LARGE-SCALE PROGRAMMING IN MARKOV DECISION PROCESSES

A THESIS

Presented to

The Faculty of the Division of Graduate Studies

by

Luis E. Contreras

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in the School of

Industrial and Systems Engineering

Georgia Institute of Technology

June 1976

LARGE-SCALE PROGRAMMING

IN MARKOV DECISION PROCESSES

Approved:

Donovan Young, Chairman

Fred E. Williams

John J. Jarvis

Ronald L. Rardin

Douglas C. Montgomery

Date Approved by Chairman May 25, 1976

ACKNOWLEDGMENTS

The help and love of many people are reflected in this work. Emilia, my wife, I thank you above all.

I thank my father and mother for wanting and giving so much.

Dr. Douglas C. Montgomery has been a special friend and mentor throughout my stay at Tech. Dr. John J. Jarvis has contributed greatly to my education. Fellow students Bruce Schmeiser and Rick Rosenthall have done more for me than I can describe.

Thanks to Dr. Fred E. Williams and Dr. Ronald L. Rardin for their help in this work.

My greatest gratitude goes to Donovan Young, the most profound influence on my intellectual life, the sort of professor who learns his students' languages, arranges for their financial support, even diets with them. I will not soon forget how much of himself he has given to me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

ABSTRACT

This thesis presents several contributions to the art of solution of large-scale Markov decision problems.

Results on expected values of randomly-timed cash flows, previously published by this author with Donovan Young, are adapted to provide the reparameterizations needed to express a semi-Markov decision problem in terms of the equivalent Markov decision problem. J. C. Totten proved that this was possible in principle; this work makes it convenient in practice. In addition to exact procedures, it is shown that sojourn-time distributions may be represented simply by their mean and variance with little error.

A "state-change" modeling procedure is developed and shown to simplify the modeling of the large class of Markov decision problems in which all decisions consist of changes of state.

Some of the statistical inference literature needed to specify Markov transition probabilities is unified and organized into a proposed data-preparation procedure.

A taxonomy is proposed to unify and compare the various solution algorithms, including new ones developed in this thesis.

A "natural" decomposition algorithm is presented that finds the sequence in which to calculate state values so that the size of the largest independent equation system solved is minimized. For problems with appreciable structure (a class that apparently includes all large-scale applications) the natural decomposition algorithm dominates all

other tested algorithms for efficiency in solving large-scale Markov decision problems in the modest size range tested (up to 100 states).

For sizes such that not all non-zero probabilities can be stored in fast-access memory, there has previously been no alternative to some form of successive approximation. An "arbitrary" decomposition algorithm is presented that allows such problems to be solved piecemeal; its efficiency is shown to be high for very structured and very unstructured problems, but low for problems with moderate structure.

A number of proposed improvements are suggested for successive approximation procedures, but are not developed fully. The most interesting of these is an extrapolation procedure ("2-V forecast acceleration") that uses an asymptotically accurate forecast based on the known behavior of White's basic successive approximation algorithm.

CHAPTER I

INTRODUCTION

Markov Decision Processes (MDP) are Markov processes with returns associated with transitions from state to state, modified by a decision maker who observes the system between transitions and takes actions to change the transition probabilities and returns. These actions depend on the observed state. The planning horizon is considered here to be infinite, and the state and action spaces countable. The objective is to find a sequence of decisions that maximizes the total discounted expected return.

Historical Background

Bellman and LaSalle [9], and Bellman and Blackwell [8] originated the study of MDP in 1949 by looking at "games of survival" as examples of multi-stage games. Later, in 1953, Shapley [110] showed that two-person stochastic games with one of the players acting as a dummy are equivalent to MDP. Bellman [6], in 1957, introduced the term "Markovian Decision Processes," giving the first explicit formulation outside the game context. The model was shown to describe many dynamic systems, but it was not successfully applied for the lack of an efficient computational method.

The year of greatest progress was 1960. Howard [55] gave the policy iteration algorithms in both discrete and continuous time, and formulated and solved his famous problems: the taxicab problem, the

baseball problem, and the automobile replacement problem. D'Epenoux [24] presented the linear programming formulation and showed that Howard's policy iteration routine corresponds precisely to block pivoting (multiple substitution of basic variables) in the linear programming problem. Progress was also reported that same year in unraveling the subtleties of the related undiscounted problem, where long-run expected return per transition is used as the criterion in place of expected discounted returns. This problem requires consideration of the ergodic chain structure of the process. Manne [84], De Ghellnick [17] and Oliver [94] formulated the single-chain problem via linear programming, showing that the analogy between linear and dynamic programming is preserved for this simple case.

In 1962 Wolfe and Dantzig [126] gave a modification of their decomposition algorithm to solve a more general undiscounted single-chain problem, where the action space is a convex set defined by linear constraints. In 1970 Denardo [21] completed the solution of the undiscounted problem by showing that Manne's single-chain linear program finds the ergodic chain that optimizes over all chains regardless of chain structure, where the decision maker includes as one of the decisions the choice of a starting state. He also showed how to embed the single-chain program in a decision procedure that efficiently solves the multi-chain problem. The practitioner need not decide in advance whether the problem is of simple or multi-chain structure, since Manne's program yields valuable information in either case; however, Manne's program is highly degenerate in the more complex cases.

Blackwell [12], in 1962, had proved the optimality of stationary strategies by considering the MDP as a sequential decision process. That same year, Derman [25] had arrived at the same result from the functional equations of dynamic programming and had independently derived the linear programming formulations.

In 1963, Jewell [59] and [60] analyzed so-called semi-Markov decision processes (Markov renewal programming), in which the sojourn time in a state is a random variable with distribution depending upon the states visited, and gave similar policy iteration algorithms.

In 1965 Blackwell [13] studied generalized Markov decision processes, in which the state and action spaces are extended to include Borel sets, and gave existence theorems for optimal policies. The same year, Brown [14] reported results on the use of dynamic programming for a finite-space discrete-time process with a finite planning horizon. Derman [30] and Maitra [82] extended the finite-space, finite-action MDP to consider the denumerable case.

In 1967 Denardo [19], applying the principle of contraction mappings, gave a "general decision process" model subsuming all the models mentioned above. Here, if the contraction and monotonicity assumptions are satisfied, the problem reduces to finding the "fixed point" which is known to exist. This effort was continued by Denardo and Mitten [23], and Karp and Held [65]. A practical consequence of their work is that this entire family of decision processes requires only one large-scale computational procedure; specifically, since each process can be represented by a Markov decision chain (MDP with discrete time), the computational methods herein developed in the chain context.

Large Scale Programming in MDP

Most of the applications reported on MDP have been on formulations, not on solution of real situations. The MDP model has been recognized for almost twenty years as a powerful modeling tool of dynamic systems, but it has had very limited use. One of the main reasons for this lack of acceptance by practitioners is the fact that the accurate formulation of many problems requires the use of a large number of states. Large problems cannot be conveniently solved by direct application of Howard's technique since it requires the repeated solution of an N×N system of simultaneous linear equations, where N is the number of states.

Most of the effort on the solution of large problems has involved the method of successive approximations. White [120], in 1963, solved the undiscounted problem using this approach; Kushner and Kleinman [73, 74], in 1968 and 1971, modified this technique using some ideas of numerical analysis to obtain faster convergence; Zaldivar and Hodgson [132], in 1975, developed extrapolation techniques for accelerating the convergence of the procedure. The method of successive approximations has gained enough acceptance to be included in at least two introductory operations research textbooks, those by Wagner [117] and the second edition of Hillier and Lieberman [52].

Norman and White [92], in 1968, proposed an approximate method using the concept of an expected state that effectively replaces the stochastic problem by a deterministic system. Its use is restricted to problems where the states correspond to quantitative levels. This method has not been very popular, since it is not very clear that one would like

to formulate a problem correctly, and then to lose its essential characteristics in gaining computational advantage. Morton [89] gave a counterexample to show that the possible percentage cost error of the Norman-White procedure is unbounded.

The first true effort in the solution and study of large problems was made by Kushner and Chen [76], in 1974, by applying the decomposition method of Dantzig-Wolfe [16] to MDP. They studied the discounted and undiscounted problems, allowing for the presence of constraints, and gave a decomposition procedure with an interpretation of the subproblems generated. An earlier study, reported in 1970 by Lasdon [77], had used the ideas of generalized linear programming to formulate the MDP model, which can also take the advantages of Dantzig-Wolfe decomposition.

Of the two general approaches to large-scale MDP problems--successive approximation and decomposition--both literatures can be considered incomplete. It was not until 1975, for example, that such a standard numerical analysis technique as successive overrelaxation was reported as being used to accelerate convergence of successive approximation in a large-scale MDP application [99]; there still remains obvious room for improvement. Regarding decomposition methods, the method of Kushner and Chen is more general and cumbersom than that required by standard MDP problems and the work reported in 1968 by Fox and Landi [40] lumps a large class of reducible states together.

The literature is also incomplete in other ways. The modeling lore is quite awkward, as can be seen by comparing the inventory model given herein with that in leading textbooks [52,117]. Statistical pro-

cedures needed in data preparation for large-scale MDP problems are not very well developed, as can be seen from the review in Chapter III.

The aim of the work reported in this thesis is to help advance the art of large-scale MDP solution. Some advances in Markov-chain modeling, principally the state-change formulation (Chapter II) and modeling procedures for commonly-encountered cost timing situations (Chapter III), are presented. The literature on MDP statistical inference is organized and reviewed in the second part of Chapter III, and a special bibliography is presented. The remainder of the thesis presents the major contributions--a "natural decomposition" procedure that greatly reduces computation load for well-structured MDP problems, an arbitrary decomposition procedure that renders large unstructured problems tractable, and some promising acceleration techniques for successive approximation.

CHAPTER II

MARKOV DECISION PROCESSES

Definitions, notation and conventions used by the thirty or so most influential authors in the field of Markov decision processes have varied quite widely. The purpose of this chapter is to establish the definitions, notation and conventions used in this thesis. Everything in this chapter is well established except for the state-change formulation (which was developed jointly by this author and two other workers), and a refinement in the proof of the fundamental equation of valued Markov chains.

General Definitions

A Markov process is a stochastic process $\{X(t), t\in T\}$ with the property that the conditional probability distribution of $X(t)$ for given values of $X(t_0), X(t_1), \ldots, X(t_n)$, $t_0<t_1<\ldots<t_n$ and $t_i\in T$, for $i=0,1,\ldots,n$, depends only on $X(t_n)$, which is the most recent value of the process. Thus

$$P\{X(t)\leq x \mid X(t_n)=x_n, X(t_{n-1})=x_{n-1}, \ldots, X(t_0)=x_0\} = P\{X(t)\leq x \mid X(t_n)=x_n\}$$

The values assumed by the process are called the states, and the set of all possible values is called the state space, denoted by S. T is called the parameter space, constituting a set of times in most applications.

Depending on the nature of the parameter space and the state space, Markov processes are usually classified as shown in Table 1.

Table 1. Classification of Markov Processes

Parameter Space \ State Space	Discrete	Continuous
Discrete	Markov chain with discrete states	Markov chain with continuous state space
Continuous	Markov process with discrete states	Markov process with continuous state space

A semi-Markov process, or Markov renewal process, is a stochastic process that moves from one state to the next with given probabilities, but the sojourn time in each state is a random variable with a probability distribution that depends upon the state.

A discrete-time Markov chain is a semi-Markov process in which every sojourn time distribution is degenerate at unit times.

A continuous-time Markov process is a semi-Markov process in which all sojourn time distributions are negative-exponential.

Classification of States and Chains

In the study of Markov chains and related decision processes, one

may classify the states in the following way: A state to which return is uncertain is *transient*. A state that is not transient is *recurrent*. There are three types of recurrent states, namely *absorbing*, *periodic*, and *aperiodic*; an absorbing state is one from which transitions to other states are impossible; a periodic state is one to which the system returns at regular intervals; an aperiodic or *ergodic* state is one to which the system will return with probability one, but after an uncertain number of transitions. This classification can be summarized in the following way:

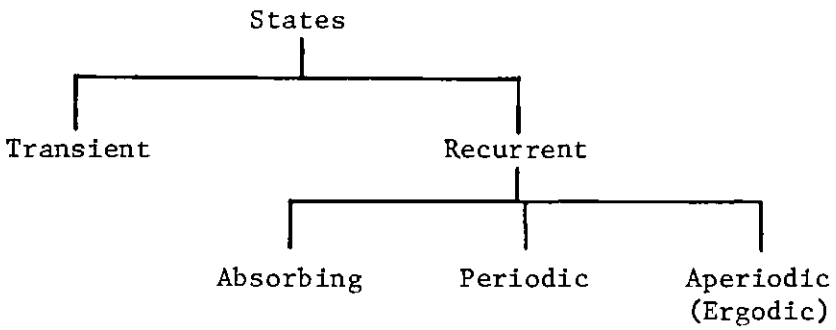


Figure 1. Classification of States and Chains

A chain can now be classified according to the type of states it contains. The underlying concept is that of *communication*. If it is possible for a system that starts in state i to reach state j, then state i is said to *communicate* with state j. If state i communicates with state j, and j with i, then states i and j *intercommunicate*. States can only intercommunicate with states of the same type. Recurrent states can be grouped into disjoint subsets within

which the states intercommunicate and outside which they do not communicate. A system with several recurrent subsets is said to be *multichain* or *polydesmic*; a system with a single recurrent set of states and possibly some transient states is said to be *unichain* or *monodesmic*. A set of states all of which intercommunicate is said to be *irreducible*. An *equivalence class* is a set of states of the same type.

The chain classification into polydesmic and monodesmic chains will be extremely useful in Chapter VI, where it is used as the basis for a proposed computational method.

Valued Markov Chains

A valued Markov chain is a Markov chain with rewards associated with visits to states, or, more generally, with transitions from state to state. It is customary to restrict attention only to stationary (time-homogeneous) valued Markov chains. Let $P\{X(t)=j \mid X(t-1)=i\}$ be denoted by the symbol $p(i,j)$ for all t, and let $P=\{p(i,j)\}$ denote a square matrix called the transition probability matrix, or simply the transition matrix. The reward associated with a transition from state i to state j is denoted by $c(i,j)$, and the square matrix $C=\{c(i,j)\}$ is the matrix of transition rewards, or simply the reward matrix. The state space S is assumed to contain a finite number of elements represented by the integers $1,2,\ldots,N$. This thesis treats only infinite-horizon valued Markov chains, in which the parameter space is $T=\{0,1,2,\ldots\}$.

Measures of Performance

The object of analyzing a valued Markov chain is assumed to be

to select among alternative chains a chain whose rewards are most favorable, as measured by some measure of performance that is a function of the rewards and their probabilities of being received. The most commonly used measure of performance for a valued Markov chain is the expected value of the present worth of the infinite series of rewards starting from a given state. An alternative measure applicable to a chain with a rewardless absorbing state is the expected total of rewards. Another alternative measure, one of particularly great historical interest, is the long-run expected reward per transition. This measure is applicable to ergodic chains only, and considers only the steady-state properties of the chain. This thesis treats only the expected present worth measure, which includes total rewards as a special case and which can approximate the long-run reward per transition to any desired accuracy.

Consider a valued Markov chain having N states numbered $1,2,\ldots,N$. Let $P.W.(i)$ denote the present worth, at discount factor β, of the infinite series of rewards to be collected when the chain starts in state $i \epsilon S$. Since the reward at any given later time is a random variable, the present worth of the infinite series of rewards is also a random variable; let $V(i)$ denote its expected value.

The fundamental relationship on which all infinite-horizon theory of valued Markov chains depends is a simultaneous set of linear equations for V_i for all i. A development of these equations that does not depend on invoking the economic interpretation of present worth, nor on invoking the intuitive idea that the present worth does not change with time for an infinite horizon, is as follows. This proof uses only the definition of present worth, the definition of expected value, and

the Markov conditional probability law.

Let $f(i,n)$ represent the reward collected at time n, given that the process starts in state i ($i \in S = 1,2,\ldots,N$) ; $n=0,1,2,\ldots$. By definition,

$$P.W.(i) = \sum_{n=0}^{\infty} f(i,n)\beta^n \quad , \qquad i=1,2,\ldots,N$$

and

$$V(i) = E\left(P.W.(i)\right)$$

$$= E\left(f(i,0)+\beta f(i,1)+\beta^2 f(i,2)+\ldots\right)$$

$$= E\left(f(i,0)\right)+\beta E\left(f(i,1)+\beta f(i,2)+\ldots\right) \quad .$$

Since this is a stationary Markov chain, the state probabilities for various states j at time n, given state i at time $n-1$, have the probability distribution $\{p(i,j)\}$, which can be used to compute the expected reward at time n :

$$E\left(f(i,n)\right) = \sum_{j=1}^{N} p(i,j)E\left(f(j,n-1)\right)$$

Thus, by recursively applying this relationship, which is true for any n, the fundamental equation for $V(i)$ follows directly:

$$V(i) = E\left(f(i,0)\right) + \beta\sum_{j=1}^{N} p(i,j)E\left(f(j,1)+f(j,2)+\ldots\right)$$

$$= E\left(f(i,0)\right) + \beta\sum_{j=1}^{N} p(i,j)V(j)$$

or

$$V(i) = r(i) + \beta\sum_{j=1}^{N} p(i,j)V(j) \qquad (1)$$

where we define the expected reward for a state visit as

$$r(i) = E\{f(i,0)\} = \sum_{j=1}^{N} p(i,j)c(i,j) \qquad (2)$$

Since $r(i)$ depends on n through $f(i,0)$, which is $f(i,n)$ for $n=0$, it remains to show whether $V(i)$ is independent of time (as is universally assumed in the literature). There are two senses in which $V(i)$ is indeed independent of time, although this aspect has apparently been ignored in the literature, possibly because of the obvious intuitive appeal of the approaches to be described later, or possibly because of the *formal* lack of dependence on time in Equations (1) and (2). The arguments to follow are intended to clear up the contradiction that arises between the usual definition of present worth with respect to a fixed time scale and the practice in the literature of giving the same name "present worth" to a quantity defined on a movable time scale.

A rigorous approach is to define $V(i)$ as the expected value of a *current worth* $S(i,n)$, where for a fixed time scale $S(i,n)$ is defined as the present worth of all rewards collected for times $t \geq n$, discounted to time n instead of to time zero (hence the name "current" worth). For rewards at times $t=n,n+1,n+2,\ldots$ the current worth of a valued infinite-horizon Markov chain is

$$S(i,n) = \sum_{t=n}^{\infty} f(i,t)\beta^{n-t} \quad .$$

With $S(i,n)$ substituted for $P.W.(i)$ in the above proof, Equation (1) remains the same except that $r(i)$ is the expected value of $f(i,n)$

rather than of $f(i,0)$. Thus, without abandoning a fixed time scale, when $V(i)$ is interpreted as an expected current worth it is indeed independent of time.

A more casual approach, implicit in the writings of the authors quoted below, is to let the present worth be defined with respect to a *movable* time scale, where $n=0,1,2,...$ refers not to fixed times but to numbers of transitions after the current time, where the current time is $n=0$. With this interpretation the development of Equations (1) and (2) stands exactly as given above, and the equations are seen to apply to any transition, not just the first transition. All present worths in this thesis will be understood to refer to the movable time scale, i.e., to be current worths. The contradiction avoided by this understanding, without resorting to $S(i,n)$ notation, is this: Let the expected present worth of being in state i at time n be $0(i,n)$ when calculated with respect to a fixed time, so that the expected present worth of being in the same state i at time $n+1$ is $0(i,n+1)=\beta 0(i,n)$ with respect to the same fixed time; the contradiction is that if these values are substituted into Equation (1), Equation (1) does not hold. We have thus established that the fundamental equation of valued Markov chains requires that present worths be calculated with respect to a movable, not fixed, time scale.

Hastings [51] has derived Equation (1) as a limiting case of the expected present worth of being in state i as the number of transitions remaining over a finite horizon becomes large. Ross [101] has given a proof involving lower and upper limits on $V(i)$ that converge to a single value. Howard [55] gave a proof of the steady-state equa-

tion that corresponds to Equation (1) when expected reward per transition is the measure of performance. Many other extant proofs, or developments, of Equation (1) are based on intuitive arguments essentially identical to the following.

Consider the stochastic activity network of Figure 2, where each node represents a given state at a given time, and each arc represents a possible transition. The chain starts in state i at time $n=0$ and makes a transition. Let $V(j)$ represent the expected (current) worth of rewards at times $n=1,2,\ldots$ if the process is in state j at time $n=1$.

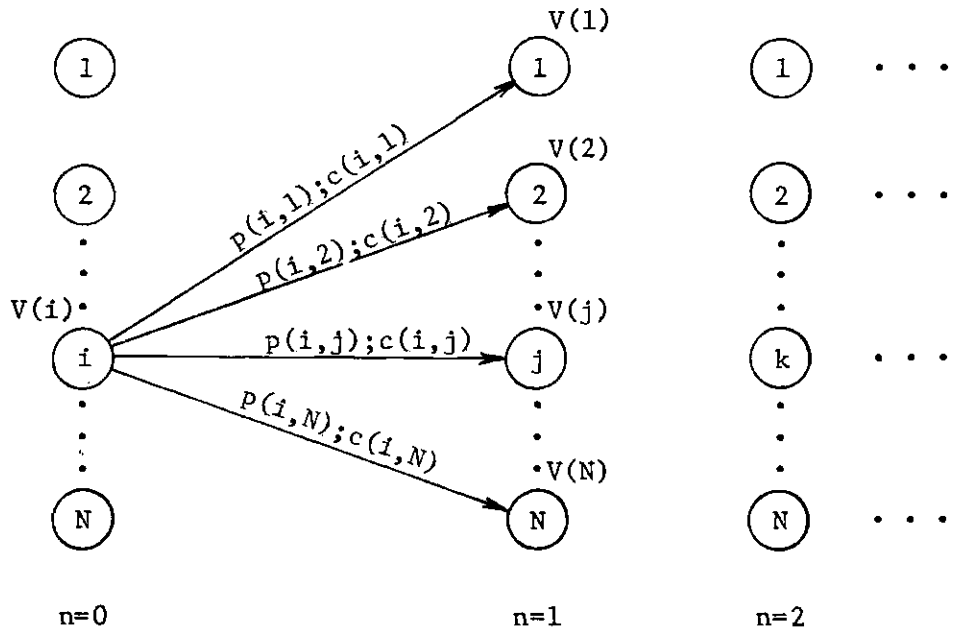


Figure 2. Stochastic Activity Network for a Valued Markov Chain

If the transition is from i to j , there will be an immediate reward of $c(i,j)$, and at time $n+1$ the expected present worth of re-

wards is $V(j)$. Thus, if the transition from i is in fact to j , we have $V(i|i\to j) = c(i,j) + \beta V(j)$. Since the visit to state j has a probability of $p(i,j)$ and there are N possible states, the present worth of starting in i is

$$V(i) = \sum_{j=1}^{N} p(i,j)\left[c(i,j) + \beta V(j)\right]$$

$$= \sum_{j=1}^{N} p(i,j)\, c(i) + \beta \sum_{j=1}^{N} p(i,j)\, c(i,j)$$

$$= r(i) + \beta \sum_{j=1}^{N} p(i,j)\, c(i,j)$$

This is, of course, Equation (1).

Totten [116] has shown that the discounted and undiscounted value equations both for a Markov and semi-Markov valued process can be reduced to Equation 1 by appropriate reparameterization. Thus Equation (1) is quite general. A semi-Markov valued process is a semi-Markov process with returns associated with transitions from state to state.

If a single measure is desired for all the starting states of a valued Markov chain, with initial probability distribution given by $\pi(i,0)$, $i=1,2,\ldots,N$, we can define the expected present worth of the chain, V , to be

$$V = \sum_{i=1}^{N} \pi(i,0)\, V(i)$$

where $\pi(i,0) \geq 0 \qquad i=1,2,\ldots,N$

$$\sum_{i=1}^{N} \pi(i,0) = 1$$

Markov Decision Processes

A Markov Decision Process (MDP) is a controlled valued Markov chain. The process is observed by a decision maker at time points $n=0,1,2,\ldots$, to be in state $i \varepsilon S$; the decision maker then chooses an action k , $k=1,2,\ldots,K_i$, which affects the conditional probabilities and rewards for the transition. The objective of the decision maker will be taken here to be the maximization of the expected present worth of the infinite stream of rewards.

Standard Formulation

The standard formulation of MDP was given by Howard [55] and has not been improved up to now. For each state $i \varepsilon S$ there are K_i versions of the ith row of transition probabilities and returns. The decision maker is viewed as choosing the best row for every state to maximize the expected present worth of the process. The classical example of this formulation is "Howard's taxicab problem" [55].

To formulate a problem using this concept, we need to define the transition probabilities and returns for every state and under every action. The problem can be stated as that of solving the functional system of equations given by

$$V(i) = \max_k \left\{ r(i,k) + \beta \sum_{j=1}^{N} p(i,j,k)\, V(j) \right\} \qquad i=1,2,\ldots,N \qquad (3)$$

where

$$r(i,k) = \sum_{j=1}^{N} p(i,j,k)\, c(i,j,k)$$

The observation, action and transition are viewed as taking place in

quick succession, all at time n. The decision variables are the values for k at each state. Note that once a policy has been given, Equation (3) is identical to Equation (1), so that the MDP problem becomes that of solving Equation (1) for a valued Markov chain. By a policy we mean a set of chosen k's, one for every state.

State-Change Formulation

The State-Change Formulation was first reported by Young [127] as a way of formulating a problem in which all of the actions are changes in state. The decision maker is viewed as observing the system at time n, $n=0,1,2,\ldots$, and immediately changing the state of the system i, to a temporary state ℓ at a cost of $k(i,\ell)$. The decision for this state change is denoted by $d(i,\ell)$, which represents the probability that the chosen temporary state will be ℓ, given that the observed state is i. After this, there is a chance move which takes the system to state j with a probability $p(\ell,j)$ and a return $c(\ell,j)$. This idea is illustrated in Figure 3.

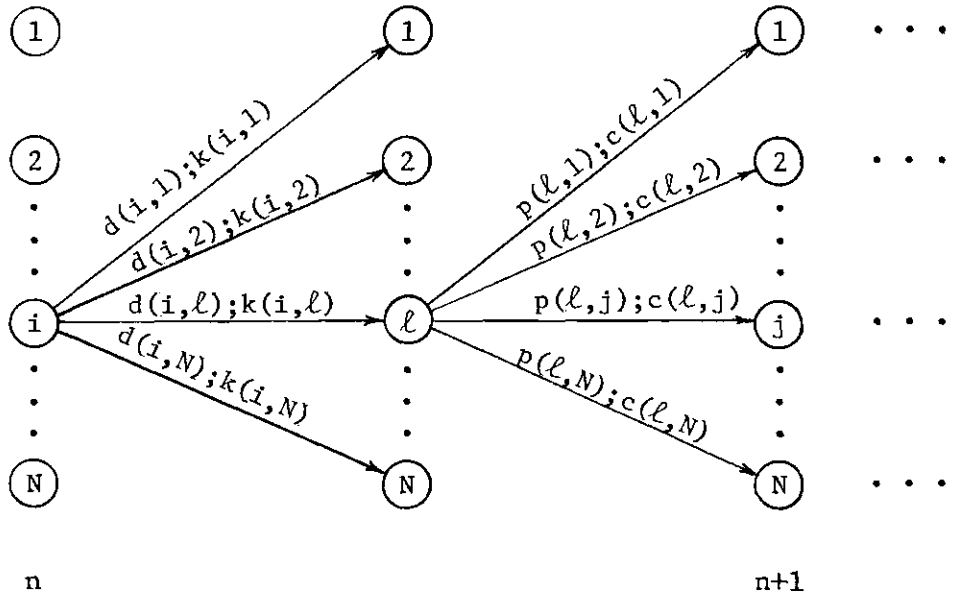


Figure 3. Stochastic Activity Network for the State-Change Formulation

Once the decision variables $d(i,\ell)$ have been specified, the state-change problem is a valued Markov chain, and we can write down Equation (1) in terms of the decision variables as follows:

1. First consider finding the expected immediate return denoted by $r'(i)$. If we make a state change from i to ℓ, we get the immediate expected return of a transition from ℓ given by $\sum_{j=1}^{N} p(\ell,j)\ c(\ell,j)$, minus the cost of the state change from i to ℓ, $k(i,\ell)$. Since we make a state-change from i to ℓ with probability $d(i,\ell)$,

$$r'(i) = \sum_{\ell=1}^{N} d(i,\ell)\left[-k(i,\ell) + \sum_{j=1}^{N} p(\ell,j)\ c(\ell,j)\right] \qquad (4)$$

2. To find the probability of going from i to j, $p'(i,j)$,

we need only consider that we will first make a state-change from i to ℓ with a probability $d(i,\ell)$, to get

$$p'(i,j) = \sum_{\ell=1}^{N} d(i,\ell)p(\ell,j) \quad , \tag{5}$$

or in matrix notation, $Q = \{p'(i,j)\} = DP$, where Q is the equivalent transition matrix. Rewriting Equation (1) and replacing $r'(i)$ and $p'(i,j)$, we get

$$\begin{aligned} V(i) &= r'(i) + \beta \sum_{j=1}^{N} p'(i,j)V(j) \\ &= \sum_{\ell=1}^{N} d(i,\ell)\left\{-k(i,\ell) + \sum_{j=1}^{N} p(\ell,j)c(\ell,j)\right\} \\ &\quad + \beta \sum_{j=1}^{N}\left(\sum_{\ell=1}^{N} d(i,\ell)p(\ell,j)\right)V(j) \\ &= \sum_{\ell=1}^{N} d(i,\ell)\left\{-k(i,\ell) + \sum_{j=1}^{N} p(\ell,j)\left(c(\ell,j)+\beta V(j)\right)\right\} \end{aligned} \tag{6}$$

This equation can be easily interpreted, since for a given decision $d(i,\ell)$, the expected present worth is given as the cost of the state change plus the expected present worth of all future rewards.

Thus the MDP in state-change formulation consists of the solution of the system of functional equations given by

$$V(i) = \underset{d(i,\ell)}{\text{Max}} \sum_{\ell=1}^{N} d(i,\ell)\{-k(i,\ell) + \sum_{j=1}^{N} p(\ell,j)\left[c(\ell,j)+\beta V(j)\right]\} \qquad (7)$$

$$i=1,2,\ldots,N$$

$$\sum_{\ell=1} d(i,\ell) = 1 \quad \text{all } i$$

$$d(i,\ell) \geq 0 \quad \text{all } i,\ell$$

Equivalence of These Formulations

To see how Equation (3) and Equation (7) are equivalent, notice that the decision variables $d(i,\ell)$ for the problem at hand will take on values 0 or 1, since Equation (7) forms a trivial linear knapsack problem in $d(i,\ell)$. Thus Equation (7) is equivalent to

$$V(i) = \underset{\ell}{\text{Max}}\{-k(i,\ell) + \sum_{j=1}^{N} p(\ell,j)\left[c(\ell,j) + \beta V(j)\right]\} \qquad (8)$$

$$i=1,\ldots,N$$

or

$$V(i) = \underset{\ell}{\text{Max}}\{-k(i,\ell) + \sum_{j=1}^{N} p(\ell,j)c(\ell,j) + \beta \sum_{j=1}^{N} p(\ell,j)V(j)\} \qquad (9)$$

$$i=1,2,\ldots,N$$

The first term, $-k(i,\ell) + \sum_{j=1}^{N} p(\ell,j)c(\ell,j) = -k(i,\ell) + r(\ell)$ in Equation (9) can be written as $r(i,\ell)$ to denote the expected immediate reward of making a state change from i to ℓ and making a chance move to some state.

The second term, $\beta\sum_{j=1}^{N} p(\ell,j)V(j)$ in Equation (9) can be written as $\beta\sum_{j=1}^{N} p(i,j,\ell)V(j)$, since it is clear that once the process reaches some

temporary state ℓ, the probability of making a transition to some state j is independent of the state i before the state-change, that is, $p(1,j,\ell) = p(2,j,\ell) = \ldots = p(N,j,\ell)$. In other words, making a transition to a temporary state ℓ has the same effect as choosing the ℓth version of the ith row of P which gives the probabilities of going from i to j. Therefore, we can write Equation (7) as

$$V(i) = \max_{\ell} \{r(i,\ell) + \beta \sum_{j=1}^{N} p(i,j,\ell)V(j)\} \qquad (10)$$

$$i=1,2,\ldots,N$$

which is identical to the functional equations defining the standard formulation of the MDP given by Equation (3).

A Comparison of These Formulations

All problems can be expressed in either form; however, the state-change formulation is preferred when all of the actions consist of changes of states. Rosenthal [99] has exploited this characteristic to study a class of stochastic location problems. The state-change formulation requires definition of new states if used to solve problems like "Howard's taxi-cab problem" [55], and should not be used in such cases.

The advantage of state-change formulation is that it allows a clear separation between the actions of the decision maker and the actions of chance whenever the action space and the state space are identical. The advantage is in modeling convenience, not in solution, although for problems with a special structure allowing solution shortcuts the clarity of state-change formulation may make the structure

easier to detect or exploit.

A disadvantage of the standard formulation is that for many problems the natural decision variables are changes of states, so that there are many identical rows for various states, which require unnecessary storage of repeated information. Additionally, the transition returns in many cases are a combination of the independent costs of choice moves and the chance move rewards. Standard formulation is, however, the most natural formulation for cases in which the decisions are not necessarily state changes.

Some Examples

In this section we provide some examples in standard and state-change formulations to illustrate these concepts.

A Maintenance-Repair Model. The following "Machine Care Problem" has been formulated by Beckmann [5] using the standard formulation of MDP as follows:

> A machine can be in either of two states, "failed" (state 1) or "operating" (state 2). If the machine is failed, the possible actions are "normal repair" (k=1) or "express repair" (k=2) . If the machine is operating, the possible actions are "no maintenance" (k=1) or "preventive maintenance" (k=2) . The related transition probabilities and rewards are as follows:

$$P = \begin{array}{c} \\ 1 \\ \\ 2 \\ \end{array} \begin{array}{l} \\ k=1: \\ k=2: \\ k=1: \\ k=2: \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{bmatrix} \begin{pmatrix} .50 & .50 \\ .33 & .67 \end{pmatrix} \\ \begin{pmatrix} .25 & .75 \\ .10 & .90 \end{pmatrix} \end{bmatrix} \end{array} \qquad C = \begin{array}{c} \\ 1 \\ \\ 2 \\ \end{array} \begin{array}{l} \\ k=1: \\ k=2: \\ k=1: \\ k=2: \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{bmatrix} \begin{pmatrix} -8 & -5 \\ -15 & -12 \end{pmatrix} \\ \begin{pmatrix} 7 & 10 \\ 5 & 8 \end{pmatrix} \end{bmatrix} \end{array}$$

Find the decision policy that maximizes the long-run expected reward per transition.

This problem is not naturally a state-change problem, since the decisions cannot be interpreted as changing the state unless additional states are defined. To illustrate the correspondence between the two formulation methods, however, the problem will be reformulated as a state-change problem and solved. The solution will be shown to be the same as the known solution.

The state space must be expanded from $S=\{i\}$ to $S'=\{(i,k')\}=\{(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\}$, where state (i,k') represents that the last observed state is i and the last decision is k'. Here $\{k'\} = \{0,1,2\}$ where $k'=0$ represents that no action has been taken since the last state observation i. In this artificial state-change format, the problem statement leads to

$$P = \begin{array}{c} \\ (1,0) \\ (1,1) \\ (1,2) \\ (2,0) \\ (2,1) \\ (2,2) \end{array} \begin{array}{c} \begin{array}{cccccc} (1,0) & (1,1) & (1,2) & (2,0) & (2,1) & (2,2) \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ .50 & 0 & 0 & .50 & 0 & 0 \\ .33 & 0 & 0 & .67 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ .25 & 0 & 0 & .75 & 0 & 0 \\ .10 & 0 & 0 & .90 & 0 & 0 \end{bmatrix} \end{array}$$

Here the (1,0) and (2,0) rows are arbitrary, since the temporary (after-decision) state cannot be (1,0) or (2,0). The state-change cost matrix K has zeroes for all legal state changes, since the

matrix C from the original problem statement includes the effects of action costs and state-occupancy rewards combined:

$$K = \begin{array}{c} \\ (1,0) \\ (1,1) \\ (1,2) \\ (2,0) \\ (2,1) \\ (2,2) \end{array} \begin{array}{c} \begin{array}{cccccc} (1,0) & (1,1) & (1,2) & (2,0) & (2,1) & (2,2) \end{array} \\ \begin{bmatrix} \infty & 0 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & 0 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix} \end{array}$$

The transition reward matrix is

$$C = \begin{array}{c} \\ (1,0) \\ (1,1) \\ (1,2) \\ (2,0) \\ (2,1) \\ (2,2) \end{array} \begin{array}{c} \begin{array}{cccccc} (1,0) & (1,1) & (1,2) & (2,0) & (2,1) & (2,2) \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -8 & 0 & 0 & -5 & 0 & 0 \\ -15 & 0 & 0 & -12 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 10 & 0 & 0 \\ 5 & 0 & 0 & 8 & 0 & 0 \end{bmatrix} \end{array}$$

This completes the state-change formulation. In general, if an arbitrary problem is to be formulated in state-change formulation, the number of states will be $N+\Sigma_i K_i$, where N is the number of states in standard formulation and K_i is the number of decisions at state i.

From closer inspection of the original data, an alternative state-change formulation can be obtained that probably matches the problem author's thinking more closely. This alternative formulation is suggested by the fact that the original transition reward matrix has constant row and column differences, which in turn suggests that the author may have been thinking in terms of an operating profit of \$4, \$7 (halfway) and \$10, respectively, for a failed period, a half-operating period and an operating period, with action costs of \$12 for normal repair, \$19 for express repair, \$0 for no maintenance and \$2 for preventive maintenance. Under these assumptions the following algebraically-equivalent state-change formulation would have P as before, and

$$K = \begin{array}{c} \\ (1,0) \\ (1,1) \\ (1,2) \\ (2,0) \\ (2,1) \\ (2,2) \end{array} \begin{array}{c} \begin{array}{cccccc} (1,0) & (1,1) & (1,2) & (2,0) & (2,1) & (2,2) \end{array} \\ \begin{bmatrix} \infty & 12 & 19 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & 2 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix} \end{array}$$

with

$$C = \begin{array}{c} \\ (1,0) \\ (1,1) \\ (1,2) \\ (2,0) \\ (2,1) \\ (2,2) \end{array} \begin{array}{c} \begin{array}{cccccc} (1,0) & (1,1) & (1,2) & (2,0) & (2,1) & (2,2) \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 7 & 0 & 0 \\ 4 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 10 & 0 & 0 \\ 7 & 0 & 0 & 10 & 0 & 0 \end{bmatrix} \end{array}$$

To verify that both of the above state-change formulations are valid representations of the original problem, the problem was solved by a computer program (the one used in the second experiment in Chapter VII) that reads its data in state-change form. For both cases, the results were (for β=.9)

Value of state (i,k')	Optimal temporary state
V(1,0) = 34.84	(1,1) (Corresponds to k=1 at state 1)
V(1,1) = $-\infty$	-
V(1,2) = $-\infty$	-
V(2,0) = 57.03	(2,2) (Corresponds to k=2 at state 2)
V(2,1) = $-\infty$	-
V(2,2) = $-\infty$	-

These solutions correspond to the known solution of the same problem in standard formulation (see, for example, the final section of Chapter IV).

An Inventory Model. A natural example of state-change formulation is provided by the most general inventory-control problem that fits the framework of Markov decision processes: the quick-replenishment, infinite-horizon inventory problem with an arbitrary stationary probability distribution of demand and with non-linear replenishment, shortage and holding costs. Here the state is the number of units on hand, and the decision is how many to replenish, so that state-change formulation is natural.

Consider an inventory system with i units on hand at the beginning of a period; the replenishment decision brings the number of units up to ℓ, at a cost $M(\ell-i)$; the demand brings the number of units down to j, according to a probability function $P(y)$, where y is the number of units demanded, so that $j = \max(\ell-y,0)$; if $j>0$ a holding cost $H(j)$ is encountered, and if $j=0$ and expected shortage cost $U(\ell)$ is encountered that equals the expected value of the shortage cost $Q(y-\ell)$ over all possible demands y that cause $j=0$:

$$U(\ell) = \sum_{y=\ell}^{\infty} P(y|y\geq\ell)Q(y-\ell) \quad .$$

The involved computations necessary to convert the data from the form here to the form needed for standard formulation are outlined in an elementary textbook [52], using about four pages of explanation in two sections to treat the modeling of a less general inventory problem (the difficulty being that each element of the standard transition and cost matrices involves the combination of the effects of replenishment

and demand). In the state-interchange formulation, however, the three matrices that contain the data specification can be written down immediately (here for a system where inventories have an upper limit of four for illustrative purposes):

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1-\Sigma & P(0) & 0 & 0 & 0 \\ 1-\Sigma & P(1) & P(0) & 0 & 0 \\ 1-\Sigma & P(2) & P(1) & P(0) & 0 \\ 1-\Sigma & P(3) & P(2) & P(1) & P(0) \end{bmatrix} \end{array}$$

$$K = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & M(1) & M(2) & M(3) & M(4) \\ \infty & 0 & M(1) & M(2) & M(3) \\ \infty & \infty & 0 & M(1) & M(2) \\ \infty & \infty & \infty & 0 & M(1) \\ \infty & \infty & \infty & \infty & 0 \end{bmatrix} \end{array}$$

$$C = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} -U(0) & & & & \\ -U(1) & -H(1) & & & \\ -U(2) & -H(1) & -H(2) & & \\ -U(3) & -H(1) & -H(2) & -H(3) & \\ -U(4) & -H(1) & -H(2) & -H(3) & -H(4) \end{bmatrix} \end{array}$$

In the notation $1-\Sigma$ in matrix P , Σ denotes the row sum excluding the element in which Σ appears. The negative signs in matrix C reflect the convention that C is a matrix of rewards. The blanks in matrix C are unspecified rewards for impossible transitions, since the corresponding elements of P are zero. The infinite values in matrix K could be left blank also if the solution algorithm is set to ignore negative replenishment decisions ($d_{i\ell}$ set to zero for $t<i$). This problem will be used to illustrate the proposed solution methods in Chapter VII, where a generalized (s,S) policy is found to be optimal for a 100-state example.

Semi-Markov Decision Processes

A semi-Markov decision process is a controlled semi-Markov valued process. The decision maker observes the process and takes actions just as in a MDP, the only difference being in the underlying stochastic behavior.

A recent article by Lippman [79] refers to semi-Markov decision processes as being "...the most natural formulation for a plethora of stochastic control problems, particularly those arising in inventory, queueing, and inspection-maintenance-replacement systems and in economic planning and consumption models."

For a given policy, the problem reduces to that of evaluating a semi-Markov valued process, which is an equivalent problem to that of a valued Markov chain, as shown by Totten [116]. Thus, the results presented in this thesis apply also for semi-Markov decision processes.

Totten does not give the conversion procedures, but shows their

existence. The detailed procedures applicable for most of the commonly encountered sojourn-time distributions are precisely those given by Young and Contreras for calculating expected present worths of randomly-timed cash flows [129]; they are elaborated and applied to stochastic processes in a forthcoming master's thesis by Carl Wohlers (School of Industrial and Systems Engineering, Georgia Institute of Technology) on decision trees with random timing.

CHAPTER III

INFORMATION GATHERING FOR MARKOV DECISION PROCESSES

The computation difficulties that are the main subject of this thesis are not the only impediments to large-scale MDP usage. The area of information gathering may be equally important. Information gathering and modeling are intimately connected, since there are tradeoffs among alternative formulation models along competing dimensions. Computational convenience, accuracy and demand for data are three of the most important such dimensions, although such considerations as generality, robustness and intuitive appeal may be equally important.

In the previous chapter the state-change formulation was introduced. It is suitable only for certain MDPs, but for these it is superior to standard formulation in several ways--computational convenience (reduced storage), reduced preprocessing of data (probabilities and costs are used separately as gathered, not lumped together) and intuitive appeal--although it has no accuracy advantage in computation and is less general. From the standpoint of information gathering, state-change formulation has an additional advantage in that, because it does not confound rewards, costs and probabilities, it allows easier and more accurate application of the inferences developed in this chapter for helping to specify rewards and probabilities.

This chapter treats two general aspects of MDP information gathering. In the first section some new cost-modeling (or reward-

modeling) methods are presented; these are direct consequences of recent work in uncertain timing of cash flows by Young and Contreras [129] applied to MDP. In the second section a framework for applying statistical inferences to large-scale MDP is proposed; the inference methods themselves are known methods from the literature and are simply explained briefly.

Cost Timing in Discrete Markov Chains

In this section the effects of random timing of rewards within a transition interval are examined, and methods are presented for estimating $c(i,j)$, or $c(\ell,j)$ and $k(i,\ell)$ when the costs or rewards are known, but their timing is uncertain.

In both the standard formulation and the state-change formulation, the value equations contain by convention the assumption that the observation, the action and the chance transition all occur at time n . The assumed timing is simply a convention. If, for example, a transition reward were collected at the end of a transition interval rather than at the beginning, we could simply replace $r(i)$ in Equation (1), Chapter II, with $\beta r(i)$.

The detailed cost timing considerations are considered in the context of a small example problem.

Consider an amplifier that can exist in 3 states, where state 3 is failure, an absorbing state. The elements of $\{p_{ij}\}$ are .92, .07, .01, 0, .55, .45, 0, 0, 1. The amplifier begins in state 1, and it is desired to find the expected number of time periods to failure. This process can be considered an infinite-horizon stationary Markov chain

with rewards if we imagine a $1 reward to be collected each time the amplifier does not fail; then the total number of dollars collected is equal to the total number of time periods to failure. Thus Equation (1), Chapter II, applies, with $\beta=1$. We set $c(3,3)=0$. Now the question of timing arises. We can compute the expected number of time periods completed before failure by setting $c(i,1)=c(i,2)=1$ for all i ; or we can compute the expected number of time periods started before failure by setting $c(1,j)=c(2,j)=1$ for all j ; or we can compute an approximation of the expected number of time periods spent without failure by setting $c(i,j)=1$ for all i,j where both i and j are not 3, and $c(i,j)=.5$ for all i,j where either $i=3, j\neq 3$ or $i\neq 3, j=3$, thus assuming that transitions occur, on the average, in the middle of the year. In a sense, all three modeling approaches are satisfactory, but in another sense not. The numerical results from Equation (1), Chapter II, are, respectively, 13.44 expected time periods completed, 14.44 expected time periods started, 13.94 expected time periods spent (the third answer is the average of the first two). Few analysts would quarrel seriously with the third approach when applied to a real problem, but, as will be seen, this approach does not generalize to more complex problems. A less ambiguous approach is needed.

Timing of State Transitions

State transitions can occur at the beginning of the transition interval. When they do, the corresponding reward $c(i,j)$ is simply the reward for the transition. If they occur at the end of the transition interval, the corresponding reward $c(i,j)$ is $R\beta$, where R is

the reward. If they occur at a specified time within the interval, the corresponding reward $c(i,j)$ is $R\beta^{\tau}$, where τ is the proportion of the transition interval that elapses before the transition. These three rules simply discount the reward properly in accordance with the convention that Equation (1), Chapter II, is written, assuming each formal reward to occur at the beginning of the interval.

The more interesting cases are those in which a transition occurs at some unspecified time during the transition interval, as was the case in the above example. Three cases are of special interest: Uniformly distributed timing, truncated-exponential timing, and arbitrarily distributed timing.

Uniformly Distributed Transition Timing. Let τ be a uniformly distributed random variable on the interval (0,1), representing the proportion of the transition interval that elapses before the transition. Let R be the actual reward collected. Then, directly from the arguments in Young and Contreras [129], we have

$$c(i,j) = R\frac{1-\beta}{-\ell n\ \beta} \quad \begin{matrix}(\textit{Transition reward } R \textit{ at} \\ \textit{uniformly distributed time.})\end{matrix} \qquad (1)$$

Uniformly distributed timing is what would be expected if the transition were the result of an event whose arrival time had a negative-exponential distribution, that is, an event from a series of events that constitutes a Poisson process. Thus this case fits many practical applications. The quantity $-\ell n\ \beta$ is of course a positive quantity known to economists as the nominal continuous interest rate r. By definition, $\beta=e^{-r}$ or $r=-\ell n\ \beta$.

Truncated-Exponential Transition Timing. If the transition were the result of an event whose timing had a negative-exponential distribution with mean m transition intervals, and the event happened to have occurred during the interval, then we would expect the timing of the transition to have a truncated-exponential distribution. This covers some important aging or reliability situations, where the event occurs on the first failure in a series of successive chances for failure. Let x be the time of the event, with density function $f(x) = (1/m)e^{-x/m}$, so that m is the expected time of the event. If $x<1$, then we set $c(i,j)$ equal to the expected value of $R\beta^x$. But this expected value is

$$E(R\beta^x|x<1) = E(Re^{-rx}|x<1) = \frac{R\int_0^1 e^{-rx}f(x)dx}{\int_0^1 f(x)dx}$$

Integrating, we obtain

$$c(i,j) = \frac{R(1-e^{-(r+1/m)})}{(1+rm)(1-e^{-1/m})} \quad \begin{matrix}(\textit{Transition reward } R \textit{ at} \\ \textit{truncated-exponential} \\ \textit{time})\end{matrix} \qquad (2)$$

Arbitrarily Distributed Transition Timing. Within the usual range of practical interest for economy studies in engineering (interest rates up to about 18 per cent, transition intervals of less than 10 yr duration), an approximation given by Young and Contreras [129] for the expected value of $e^{-r\tau}$ (which is the same as β^τ here) leads to the following approximation for $c(i,j)$ when the time τ of the transition is distributed arbitrarily with mean μ and variance σ^2 :

$$c(i,j) = R\beta^{\mu}(1+\sigma^2 r^2/2) \qquad \begin{matrix}\text{(Transition reward } R \text{ at} \\ \text{arbitrarily distributed} \\ \text{time)}\end{matrix} \qquad (3)$$

The approximation of Equation (3) holds within about 1 per cent error for practically any probability distribution of τ .

The three formulas given for $c(i,j)$ in Equations (1), (2) and (3) should cover most cases of practical interest where a transition reward is collected at the moment of transition, and this moment is randomly distributed.

Cessation of Uniform Rewards

We now consider situations in which the reward is not a lump sum paid at the moment of transition, but is a uniform cash flow that starts or stops at the moment of transition. If state i is a state in which we collect $\bar{A}$ dollars per unit time, and state j is a state in which we collect no reward, then $c(i,j)$ should be set to the expected value of $\bar{A}(1-\beta^{\tau})/r$ in order to have $c(i,j)$ represent the expected present worth of earnings up to cessation. Conversely, if state i is a state in which we collect no reward, and state j is a state in which we collect $\bar{A}$ dollars per unit time, then $c(i,j)$ should be set to the expected value of $\bar{A}(\beta-\beta^{\tau})/r$. These expected values can be calculated from the results in the previous section.

Table 2 summarizes the formulas for $c(i,j)$ resulting from consideration of lump sums or uniform cash flows with random starting or cessation times within a transition interval.

Table 2. Contributions to $c(i,j)$
for Single and Uniform Continuous Rewards

Single Reward

R dollars at time τ

Timing of Rewards	Contributions to $c(i,j)$	
τ a known constant	$R\beta^{\tau}$	
(τ=0)	(R)	
(τ=1)	($R\beta$)	
τ distributed uniformly (0,1)	$R(1-\beta)/r$	($r=-\ell n\ \beta$)
x distributed exponentially with mean μ, giving truncated-exponential τ	$\frac{R(1-e^{-(r+1/m)})}{(1+rm)(1-e^{-1/m})}$	(i,j such that x<1)
τ distributed arbitrarily with mean μ, variance σ^2	$R\beta^{\mu}(1+\sigma^2r^2/2)$	

Uniform Continuous Reward

$\bar{A}$ dollars per unit time from 0 to τ,

$\bar{B}$ dollars per unit time from τ to 1

Timing of Rewards	Contributions to $c(i,j)$
$\bar{A}$, time 0 to time τ	
τ a known constant	$\bar{A}(1-\beta^{\tau})/r$
τ distributed uniformly (0,1)	$\bar{A}(1-(1-\beta/r))/r$
τ distributed arbitrarily with mean μ, variance σ^2	$\bar{A}(1-\beta^{\mu}(1+\sigma^2r^2/2))/r$
$\bar{B}$, time τ to time 1	
τ a known constant	$\bar{B}(\beta-\beta^{\tau})/r$
τ distributed uniformly (0,1)	$\bar{B}(\beta-(1-\beta/r))/r$
τ distributed arbitrarily with mean μ, variance σ^2	$\bar{B}(1-\beta^{\mu}(1+\sigma^2r^2/2))/r$

Statistical Inference

In practical situations, the gathering of MDP data is not trivial. For example, it may be desired to apply MDP to a situation that does not necessarily obey the Markov assumptions when reasoned from first principles, so that sample trajectories from real or simulated histories of the process must be examined to test the relevant hypotheses. Such a situation occurs when states are lumped together, as when inventory control is done in lots of M items, when insured clients are classified into risk classes, when queueing systems are examined at regular intervals whose lengths are such that many state changes occur between observations, and when continuous variables are treated as discrete. Such a situation also occurs when artificial states are defined so as to represent a non-stationary chain with a stationary one, as when actuaries devise states intermediate between "living" and "dead" in order to replicate observed survivorship curves, or when "incubation" states are added in disease control problems. In addition, there are cases where micro knowledge is great but macro knowledge is small, as when the overall operation of a baggage handling system or a production control system is hypothesized to have Markov properties. Finally, there are many cases in which one wonders whether Markov assumptions apply at all, as in marketing surveys where only experimental evidence would be able to establish that brand switching can be adequately modeled as Markovian.

Even with enough data to establish a set of observable states that obey the Markov assumptions, it is doubtful that anyone with a large-scale problem to solve would have information first-hand on all

of the possible versions of the Markov chain that could be encountered when many different decision policies are tried. Thus we postulate that at least some of the data gathering will be done be simulation. Once a decision situation becomes well known in one or more versions, as by actual history, one may build a simulation model of it. If the simulation model represents the known versions of the situation, there is reasonable hope that other versions of the simulation model would also represent versions of the situation that exist so far only in the model builder's imagination. The method that is probably most typically used in simulation studies is simply to try various decisions in a validated simulation model, choosing the decisions that give the best outcome.

Where the problem has MDP structure, it seems obvious that a much more efficient way to proceed is to use the simulation model only to generate the data for MDP solution, because the MDP methodology can investigate far more policies far more cheaply than direct simulation.

A recommended procedure for organizing the data investigation phase of an MDP study is shown in Figure 4. In every case where simulation is mentioned, it is of course possible and preferable to use actual histories if available.

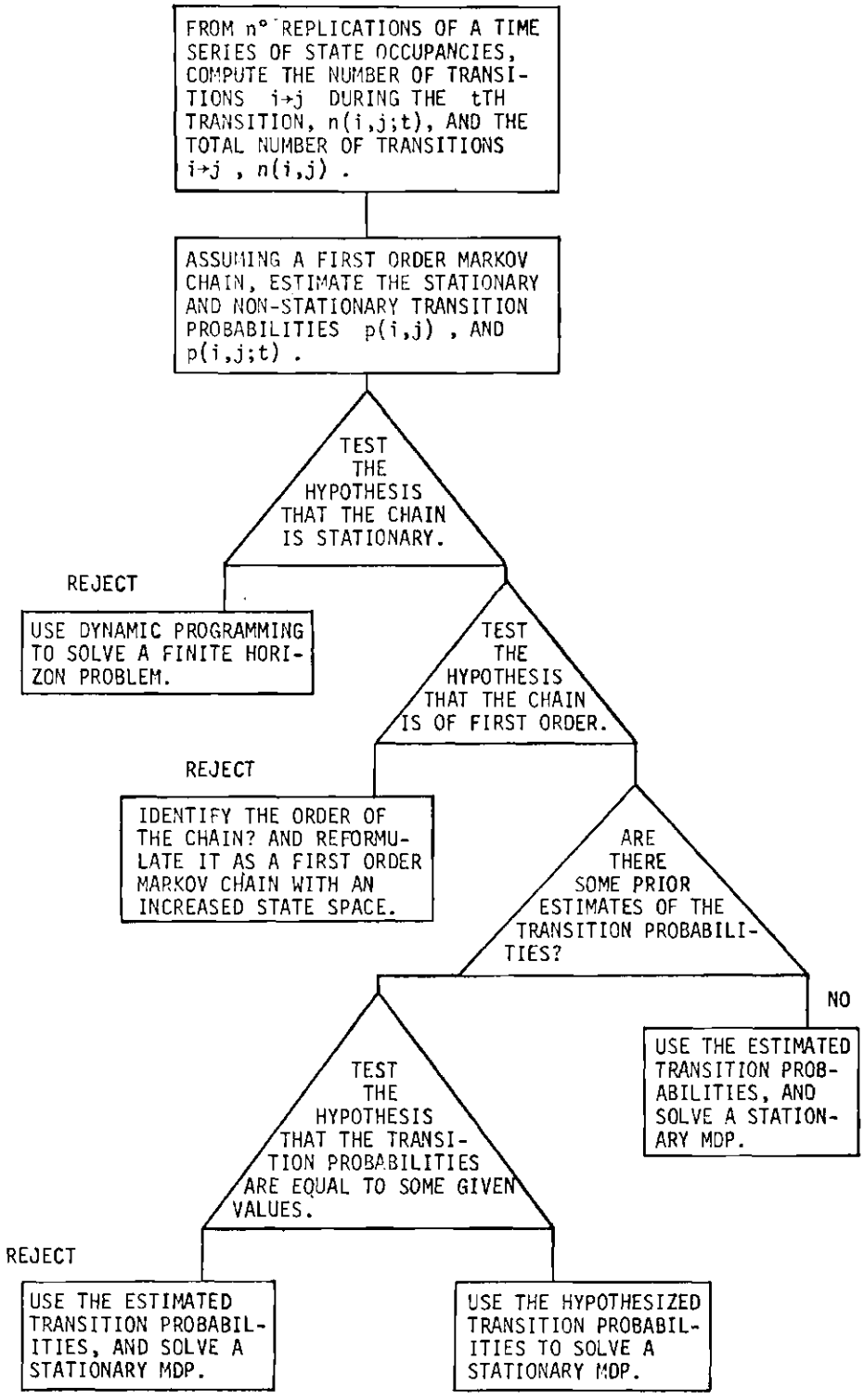


Figure 4. Data Investigation Procedure

Data Investigation

For a given state i, the probability of a transition $i \to j$ depends only on the decision at i and not on the decisions at any other state. Thus there is no need, either in simulation or in examination of historical data, to consider all policies separately. The minimal number of separate experiments necessary to estimate all possible $p(i,j,k)$ is $\Sigma_i K_i$, although for a given MDP it is usually necessary to augment this number in order to achieve a sufficient sample size for each i; if a state is never or seldom reached under a particular policy, which often happens, experimentation with other policies is required.

For each policy a time series of states resulting from a sample realization is generated. This time series is called a sample trajectory. We consider two situations: one where the time series data are available, and one where the data are aggregated in such a way that the time relationships are lost. To illustrate, time series data may be reported as a vector such as (1,1,3,4,1,1,1,3,...), meaning that the process moved from state 1 to state 1 to state 3 to state 4 . . . , where the dimension of the vector is the number of time periods observed. By contrast, aggregate data may be reported as a vector such as (36,24,56,...), meaning that state 1 was visited 36 times, state 2 was visited 24 times, . . . , where the dimension of the vector is N, the number of states.

The problem of statistical inference is of course simpler for time series data than for aggregate data. Simulation-generated data can always be in time series form, but historical records are often

aggregated.

In the following two sections, titled Investigation of Time Series Data and Investigation of Aggregate Data, we present relevant results from the literature, omitting all derivations. A Short Bibliography of Data Investigation for MDP is included at the end of the chapter, and it contains references for all statistical procedures discussed here.

Investigation of Time Series Data. Time series data may be investigated to determine whether the chain is stationary or non-stationary, to determine order of the chain, to estimate transition probabilities and to test hypotheses as to the values of given elements in the Markov transition matrix. First, we present various statistical tests available when the chain is assumed to be stationary; then we present the tests available when this assumption is relaxed, including a test for stationarity itself.

Assume a stationary chain, which is one for which $p(i,j;t)=p(i,j)$ for all t under a given policy. In a finite stationary chain with N states, let n be the number of transitions in the time series sample, let $n(i,j)$ be the number of transitions from state i to state j, and let $n(i)=\Sigma_j n(i,j)$ be the total number of transitions from state i to any state j, including $j=i$.

A maximum likelihood estimator for $p(i,j)$ for a given state i is easily derived from the sample transition counts $\{n(i,1),n(i,2),\ldots,n(i,N)\}$, which can be considered as a sample of size $n(i)$ from a multinomial distribution with probabilities $\{p(i,1),p(i,2),\ldots,p(i,N)\}$. The estimator is simply the transition frequency:

$$\hat{p}(i,j) = \frac{n(i,j)}{n(i)} \tag{4}$$

Kendall and Anderson and Goodman show that these estimators are asymptotically normally distributed. They are consistent but not generally unbiased, although the bias tends to zero as the sample size increases.

Convenient hypothesis tests for estimators of $p(i,j)$ follow directly from the asymptotically normal distribution of $\hat{p}(i,j)$. The statistic $\sqrt{n(i)}\left(\hat{p}(i,j)-p(i,j)\right)$ has an asymptotic normal distribution with mean 0 and variance $p(i,j)\left(1-p(i,j)\right)$, and thus constitutes a goodness-of-fit statistic that can be used as a basis for the following tests.

Let $\underline{P}(i)$ denote the ith row of P. Where $\underline{P}^0(i)$ is some specified row that may be different from the ith row of the maximum likelihood estimators $\{\hat{p}(i,j)\}$, the hypothesis $H_0:\underline{P}(i)=\underline{P}^0(i)$ is tested in terms of the statistic

$$S(i) = \sum_{j=1}^{N} \frac{n(i)\{\hat{p}(i,j)-p^0(i,j)\}^2}{p^0(i,j)} ,$$

where the sum is computed over non-zero elements of $\underline{P}^0(i)$ only. Let d be the number of zero elements, so that the sum contains $N-d$ non-zero elements. Then the statistic $S(i)$ is distributed χ^2 with $(N-1)-d$ degrees of freedom. H_0 is rejected if $S(i)>\chi^2_\alpha$ for the given number of degrees of freedom and a given type I error probability α.

If $\underline{P}^0(i)$ has no zero elements, an alternative test of the same hypothesis follows from a test statistic $S'(i)$ obtained from the

likelihood ratio:

$$S'(i) = 2\sum_{j=1}^{N} n(i,j)\ell n\frac{n(i,j)}{n(i)p^0(i,j)} = 2\sum_{j=1}^{N} n(i,j)\left(\ell n\hat{p}(i,j)-\ell n p^0(i,j)\right)$$

$S'(i)$ is distributed χ^2 with $N-1$ degrees of freedom, so that H_0 is rejected if $S'(i)>\chi^2_\alpha$ for the given number of degrees of freedom and a given type I error probability α.

Extension of the above two hypothesis tests on individual rows of P to the equivalent tests on the entire transition matrix P is straightforward. Let P^0 denote some specified transition matrix that may be different from the matrix of maximum likelihood estimators $\{\hat{p}(i,j)\}$. The hypothesis $H_0:P=P^0$ may be tested in terms of the statistic

$$S = \sum_{i=1}^{N} S(i) = \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{n(i)\{\hat{p}(i,j)-p^0(i,j)\}^2}{p^0(i,j)}$$

where the sum is computed over (i,j) for which $p^0(i,j)>0$ only. Where d now represents the number of zero elements of the entire matrix P^0, the statistic S is distributed χ^2 with $N(N-1)-d$ degrees of freedom. The hypothesis $H_0:P=P^0$ is rejected if $S>\chi^2_\alpha$ for the given number of degrees of freedom and a given type I error probability α.

If P^0 has no zero elements, the alternative test statistic $S' = \sum_{i=1}^{N} S'(i)$, where $S'(i)$ is as defined in the equation above, is distributed χ^2 with $N(N-1)$ degrees of freedom, and $H_0:P=P^0$ is

rejected if $S'>\chi^2_\alpha$ for the given number of degrees of freedom and a given type I error probability α .

Some hypothesis tests on the order of a stationary Markov chain are described in the following paragraphs. In the light of the results to be presented later in this thesis, these tests take on a new importance. The natural-decomposition method SMDP allows particularly efficient solution of large first-order chains that have been generated from smaller second-order chains, so that identifying a chain as second-order need not prevent its being solved.

A Markov chain is of order n $(n=0,1,...)$ if the earliest state occupancy relevant in calculating $P\{X_t=x_t\}$ is x_{t-n} . Thus a zeroth-order Markov chain is simply an independent process, where the state occupancies are independent of each other. A first-order Markov chain is the chain discussed throughout this thesis under the unmodified name "Markov chain." A second-order Markov chain is one for which $P\{X_t=x_t|x_{t-1},x_{t-2},...\} = P\{X_t=x_t|x_{t-1},x_{t-2}\}$.

Every second-order Markov chain can be expressed as a first-order Markov chain with a state space equal to the cartesian product of state space of the second-order chain and itself. For example, a second-order Markov chain with state space given by $S_2=\{1,2\}$ can be expressed by a first-order Markov chain with state space $S_1 = S_2 \times S_2 = \{(1,1),(1,2),(2,1),(2,2)\}$. Here the state descriptor (i,j) denotes that $X_{t-2}=i$ and $X_{t-1}=j$. The first-order transition probability matrix for the first-order chain equivalent to a given second-order chain contains elements $p(i,j,k)$, each representing the probability of state (j,k) given state (i,j) one period earlier. For the

example given above, we have

$$P = \begin{array}{c} \\ (1,1) \\ (1,2) \\ (2,1) \\ (2,2) \end{array} \begin{array}{c} \begin{array}{cccc} (1,1) & (1,2) & (2,1) & (2,2) \end{array} \\ \begin{bmatrix} p(1,1,1) & p(1,1,2) & 0 & 0 \\ 0 & 0 & p(1,2,1) & p(1,2,2) \\ p(2,1,1) & p(2,1,2) & 0 & 0 \\ 0 & 0 & p(2,2,1) & p(2,2,2) \end{bmatrix} \end{array}$$

To test the hypothesis that the transition probabilities from a given state j are adequately represented by first-order transition probabilities, as opposed to second-order transition probabilities, i.e., to test H_0: $p(i,j,k) = p(j,k)$, the first step is to obtain the first-order maximum likelihood estimators $\hat{p}(j,k)$ from Equation (4) and the second-order maximum likelihood estimators

$$\hat{p}(i,j,k) = \frac{n(i,j,k)}{\sum_{\ell=1}^{N} n(i,j,\ell)} ,$$

where $n(i,j,\ell)$ represents the number of $i \to j \to \ell$ transitions in the sample. The test statistic $S(j)$,

$$S(j) = \sum_{i=1}^{N} \sum_{k=1}^{N} n^*(i,j)\left(\hat{p}(i,j,k)-\hat{p}(j,k)\right)^2/\hat{p}(j,k) ,$$

where

$$n^*(i,j) = \sum_{k=1}^{N} n(i,j,k) ,$$

is distributed χ^2 with $(N-1)^2$ degrees of freedom. The hypothesis that a first-order chain adequately represents the data is rejected if $S(j)>\chi^2_\alpha$ for the given number of degrees of freedom and a given type I error probability α.

To test the same hypothesis for an entire chain, we use the test statistic $S=\Sigma_j S(j)$, which is distributed χ^2 with $N(N-1)^2$ degrees of freedom.

All of the above statistical procedures (and, for that matter, all of the methods used in this thesis) assume stationary Markov chains, which are those for which $p(i,j)=p(i,j;t)$ for all t. Questions of stationarity raise thorny issues in Markov decision processes. It is difficult to imagine a real-world stochastic process that is truly stationary, yet many are approximately so. MDP solutions have some degree of robustness against non-stationarity, partly through the operation of the discount factor β, and partly through the desire of decision makers to have a rational basis for a policy even if it is recognized that conditions will change (one may regard an optimal policy calculated by MDP methods as the best thing to do in the absence of better knowledge about the future than is available by assuming the current probabilities will continue). Although no methods are known to this author to provide a formal basis for using knowledge about the degree of stationarity exhibited by a situation, it is obviously desirable to have stationarity information available to a decision maker in a given situation if for no other reason than to aid in his application of intuitive judgement. In the following paragraphs we present a short summary of the best-known method for judging the stationarity of a

Markov chain.

Since each time series gives only one sample of $p(i,j;t)$ for a given t, it is necessary to have n^0 replicates of the time series of state occupancies. For a given i, j and t, let $n(i,j;t)$ represent the count of the number of times state j is observed at time t when state i was observed at time $t-1$; also let $n(i;t-1)$ represent the number of times state i was observed at time $t-1$. The maximum likelihood estimator of $p(i,j;t)$ is simply the frequency

$$\hat{p}(i,j;t) = \frac{n(i,j;t)}{n(i;t-1)}$$

These estimators may be used to test the stationarity of the chain.

To test the stationarity of a given row of P, say row i, the sample test statistic $S(i)$ is suggested by

$$S(i) = \sum_{t=1}^{T} \sum_{j=1}^{N} n(i;t-1)\left[\hat{p}(i,j;t)-\hat{p}(i,j)\right]^2/\hat{p}(i,j) \quad ,$$

which is distributed χ^2 with $(N-1)(T-1)$ degrees of freedom. Thus the hypothesis of stationarity, H_0: $p(i,j;t) = p(i,j)$, is rejected if $S(i)>\chi^2_\alpha$ for the given number of degrees of freedom and a given type I error probability α. To test the stationarity of an entire transition probability matrix P, the same hypothesis is tested with the statistic $S'=\Sigma_i S(i)$, which is distributed χ^2 with $N(N-1)(T-1)$ degrees of freedom. Ad-hoc judgement is required to select α for the test.

Investigation of Aggregate Data. All statistical procedures described in the above subsection have assumed the availability of time series of state occupancies. If the data are aggregated so that the state occupancy counts $n(j;t) = \Sigma_i n(i,j;t)$ are available but the transition counts $n(i,j;t)$ cannot be recovered, the procedures suggested by Miller and extended by Lee are applicable for estimating transition probabilities.

Following Miller, let us define $q(j;t)$ as the probability of observing state j at time t, so that the Markov transition law gives

$$q(j;t) = \sum_{i=1}^{N} q(i;t-1)p(i,j) \quad . \tag{5}$$

If the unconditional probabilities $q(j;t)$ are replaced by the corresponding observed frequencies $y(j;t)$, then there will be in general no set of $p(i,j)$'s for which Equation (5) holds with probability one. To have an identity, an error term $\mu(j;t)$ is introduced so that

$$y(j;t) = \sum_{i=1}^{N} y(i;t-1)p(i,j) + \mu(j;t) \quad ,$$

or, in matrix form,

$$Y(j) = X(j)P(j) + \underline{\mu}(j) \quad , \; j=1,2,\ldots,N$$

The following assumptions are made about $\underline{\mu}(j)$:

1. $E\{\underline{\mu}(j)\} = \underline{0}$
2. $E\{\underline{\mu}(j)\underline{\mu}'(j)\} = \sigma(j)W$, where W is a $T \times T$ positive-definite diagonal matrix.

The entire set of equations is, in matrix form, $Y = XP + \underline{\mu}$, with the assumptions

1. $E(\underline{\mu}) = \underline{0}$
2. $E(\underline{\mu}\underline{\mu}') = M$, where M is a $TN \times TN$ non-diagonal singular matrix.

Using the method of least squares on this linear statistical model, with the assumption that $T>N$, Miller gives the least-squares estimators of P as

$$\hat{P} = (X'X)^{-1}X'Y \quad .$$

Miller's estimators always satisfy the condition that the $\hat{p}$'s sum to one in each row, but there is nothing in his model to guarantee that every $\hat{p}(i,j)$ falls in the interval $(0,1)$. Lee shows that Miller's method does in fact often give negative estimates of some of the probabilities when applied to reasonable test problems, and he presents an extension that restricts the estimates to be non-negative. Lee's extension produces a quadratic programming problem (where Miller's problem was linear), and the problem is amenable to solution by the quadratic programming algorithm developed by Wolfe [126].

A Short Bibliography of Data Investigation for MDP

1. Anderson, TW and LA Goodman "Statistical Inference about Markov Chains" Annals of Mathematical Statistics Vol 28, No. 1, 89-100 (1957)

2. Bhat, UN Elements of Applied Stochastic Process, John Wiley & Sons, New York (1972)

3. Blight, BJN and JL Devore "The Recursive Estimation of a Markov Chain" Journal of Applied Probability Vol 11, 394-400 (1974)

4. Brooks, DM and CT Leondes "Use of Isolated Forecasts in Markov Decision Process with Imperfect Information" AIIE Transactions Vol 6, 244-51 (1974)

5. Çinlar, E Introduction to Stochastic Process, Prentice-Hall, Englewood Cliffs, New Jersey (1975)

6. Duncan, GT and GL Lizbie "Inference for Markov Chains Having Stochastic Entry and Exit" Journal of the American Statistical Association Vol 67, 761-7 (1972)

7. Ezzati, A "Forecasting Market Shares of Alternative Home-Heating Units by Markov Processes Using Transition Probabilities Estimated From Aggregate Time Series Data" Management Science Vol 21, 462-73 (1974)

8. Howard, RA "Stochastic Process Models of Consumer Behavior" Journal of Advertising Research Vol 35-42 (1963)

9. Lee, TC, Judge, GC and RL Cain "A Sampling Study of the Properties of Estimators of Transition Probabilities" Management Science Vol 15, 374-98 (1969)

10. Lee, TC, Judge, GC and A Zellner Estimating the Parameters of the Markov Probability Model from Aggregate Time Series Data, North-Holland Publishing Company, Amsterdam (1970)

11. Madansky, A "Least Squares Estimation in Finite Markov Process" Psychometrika Vol 24, 137-44 (1959)

12. Martin, JJ Bayesian Decision Problems and Markov Chains, John Wiley & Sons, New York (1967)

13. Massy, WF, Montgomery, DB and DG Morrison Stochastic Models of Buying Behavior, The MIT Press, Cambridge, Massachusetts (1970)

14. Miller, GA "Finite Markov Process in Psychology" Psychometrika Vol 17, 149-67 (1952)

15. Miller, GA and FC Frick "Statistical Beharioristics and Sequences of Responses" Psychological Review Vol 56 (1949)

16. Naylor, TH Computer Simulation Experiments with Models of Economic Systems, John Wiley & Sons, New York (1971)

17. Telser, GL "Least Squares Estimates of Transition Probabilities" Measurements in Economics (Christ, et al, ed.), Stanford University Press, 270-92 (1963)

CHAPTER IV

MICRO SOLUTION PROCEDURES

The aim of this chapter is to demonstrate how all of the important MDP solution procedures are very closely related to each other in a way not heretofore made evident in the literature; and, in so doing, to reveal further promising solution procedures of the same class.

Howard's algorithm [55], Hastings' algorithm [49], the solution procedure of Kushner and Kleinman [73], White's method of successive approximations [120], and many other solution procedures (including the procedures developed in this thesis) can all be viewed as variations that work directly on the dual of a linear programming formulation similar to one first proposed by D'Epenoux [24] in 1963. A computer program is given here to illustrate some of the computational details of these procedures, and a taxonomy of solution procedures is presented and discussed. We begin with a brief description of the linear program that ties the various procedures together.

A Linear Programming Formulation

In an infinite-horizon Markov decision process, let $\pi(i;n)$ represent the probability that the process is in state i at time n; let $\lambda(j,k;n)$ represent the joint probability that the observer will observe state j and take action k at time n; and let $x(j,k)$ represent the z-transform, $z \equiv \beta$, of $\lambda(j,k;n)$, so that $x(j,k) = \sum_{n=0}^{\infty} \lambda(j,k;n)\beta^n$. For given $r(j,k)$ as defined previously the expected

rewards for state j at time n are $\sum_{k\in K_j} r(j,k)\lambda(j,k;n)$, $n=0,1,\ldots$, so that the expected reward at time n is $\sum_{j=1}^{N} \sum_{k\in K_j} r(j,k)\lambda(j,k;n)$, $n=0,1,\ldots$.

The expected present worth of all rewards is to be maximized:

$$\text{Maximize} \quad \sum_{n=0}^{\infty} \beta^n \sum_{j=1}^{N} \sum_{k\in K_j} r(j,k)\lambda(j,k;n)$$

Because of the Markov property, the marginal probability of being in state j at time n is calculated from the marginal probabilities of being in states i at time $n-1$:

$$\sum_{k\in K_j} \lambda(j,k;n) = \sum_{i=1}^{N} \sum_{k\in K_i} p(i,j,k)\lambda(i,k;n-1) \quad , n=1,2,\ldots$$

and

$$\sum_{k\in K_j} \lambda(j,k;0) = \pi(j;0) \quad , n=0$$

The steps to tollow--those of replacing groups of constraints and groups of variables by weighted sums--are justifiable without appeal to the fact that the resulting formulation can be shown to be identical to accepted formulations. However, here it is expedient to appeal to this fact [101,pp 150-2] and proceed directly to the formulation.

The above constraints are each multiplied by β^n and summed over n for each j , yielding a finite set of N constraints:

$$\sum_{n=0}^{\infty} \sum_{k\in K_j} \lambda(j,k;n)\beta^n = \pi(j;0) + \sum_{n=1}^{\infty} \sum_{i=1}^{N} \sum_{k\in K_i} p(i,j,k)\lambda(i,k;n-1)\beta^n$$

$$= \pi(j;0) + \sum_{n=0}^{\infty} \sum_{i=1}^{N} \sum_{k\in K_i} p(i,j,k)\lambda(i,k;n)\beta^{n+1} \quad .$$

Taking the z-transforms of the objective function and all constraints yields a linear programming formulation expressed in terms of the non-negative variables $x(j,k)$ defined earlier:

$$\text{Maximize} \quad \sum_{j=1}^{N} \sum_{k\in K_j} r(j,k)x(j,k) \tag{1}$$

$$\text{subject to} \quad \sum_{k\in K_j} x(j,k) = \pi(j;0) + \beta \sum_{i=1}^{N} \sum_{k\in K_i} p(i,j,k)x(i,k) \quad ,$$

$$j=1,2,\ldots,N \tag{2}$$

$$\text{and} \quad x(j,k) \geq 0 \qquad j=1,2,\ldots,N \; ; \; k\in K_j \tag{3}$$

As several authors have pointed out [24,18], every basic feasible solution to the linear program defined by Equations (1), (2) and (3) has the property that for each j, exactly one k exists for which $x(j,k)>0$. The set of constraints associated with $x(j,\cdot)$ will be called the j-group. This property is crucial to the efficiency of all the important MDP solution procedures, and we digress very briefly to develop it.

As in any linear program with N constraints, every basic feasible solution will have at most N non-zero variables $x(j,k)$.

On the other hand, ignoring any states that cannot be reached, if there must be at least one non-zero $x(j,k)$ for every j, then the property must hold. But for state j to be reached, $\lambda(j,k;n)>0$ for at least one k and n for j, so that the non-negative weighted sums $x(j,k)$ must be non-zero for at least one k, thus forcing the property to hold for any "well formulated" problem, i.e., one that includes no unreachable states.

As a consequence of the property that exactly one k exists, for each j, for which $x(j,k)>0$, every basic solution for which the property holds is a basic feasible solution and every basic solution for which the property does not hold is infeasible. The latter is obvious; and for the former not to hold would contradict that all pure, stationary policies are feasible.

The only solutions of interest, then, are those in which one variable $x(j,k)$ from each j-group is basic. An immediate starting basic feasible solution can be obtained by making one variable from each j-group basic. The simplex optimality test will indicate any variables whose entry into the basis will improve the solution. The simplex ratio test is unnecessary since any new basic solution with one basic variable from each j-group is automatically feasible. When any variable is added to the basis, the basic variable from the same j-group is removed from the basis. Pivots can be done singly (simplex algorithm), or all indicated pivots can be done in one operation (block pivoting or multiple basis entry). This procedure will be denoted in Chapter VI as the h-algorithm.

Dual Formulation

If we let $V(i)$ be the dual variable corresponding to the ith constraint, the dual of the linear programming formulation is

$$\text{Minimize} \quad \sum_{i=1}^{N} \pi(i;0)V(i) \tag{4}$$

$$\text{subject to} \quad V(i) - \beta \sum_{j=1}^{N} p(i,j,k)V(j) \geq r(i,k) \quad ,$$

$$i=1,2,\ldots,N \ ; \ k=1,2,\ldots,K(i) \quad , \tag{5}$$

where $K(i)$ is the number of decisions in the set K_i .

Note that the dual variables $V(i)$ are unconstrained in sign, and that the number of constraints is $K(1)+K(2)+\ldots+K(N)$. The set of constraints associated with $V(i)$ is called the i-group.

Every statement in the following paragraph is the dual equivalent of the corresponding statement in the paragraph preceding this subsection.

The only solutions of interest are those in which one constraint from each i-group is tight (equality). An immediate starting primal-feasible solution can be obtained by making one constraint from each i-group tight. The dual-simplex feasibility test will indicate any violated constraints. The dual-simplex ratio test is unnecessary since any new basic solution with one constraint tight from each i-group is automatically primal-feasible. When any constraint is made tight, the tight constraint from the same i-group is relaxed. Pivots can be done singly (dual-simplex algorithm), or all indicated pivots can be done in one operation (block pivoting or multiple basis entry).

Micro Solution Procedures

The solution methods discussed in this chapter are called "micro" because they are methods used repeatedly in the solution of the sub-problems generated by the macro methods of later chapters. These micro solution procedures may all be viewed as procedures for solving Equation (7) developed below, where Equation (6) is viewed simply as a special case of Equation (7) where maximization is not done.

We define a value iteration as the act of substituting a given set of $V(i)$ values into the right-hand side of each of the N equations represented by Equation (6), yielding a new set of $V(i)$ values. We define a policy iteration as the act of substituting a given set of $V(i)$ values into the right-hand side of each of the N equations represented by Equation (7), yielding a new set of $V(i)$ values each of which is maximal over the set of decisions $k \in K_i$ for its i, and also yielding for each i the chosen decision. The list of decisions from a policy iteration is called the current policy. Since Equations (6) and (7) differ only in the maximization operator, the only difference between a policy iteration and a value iteration is that in the latter the policy is fixed beforehand.

Howard's Algorithm

The block pivoting or multiple basis entry optimization procedure outlined above following the dual formulation of Equations (5) and (6) may be particularized as follows:

1. (Initialization) For each state i, choose an action k and write the kth version of the constraint in each

i-group as an equality:

$$V(i) = r(i,k) - \beta \sum_{j=1}^{N} p(i,j,k)V(j) \quad , i=1,2,\ldots,N \quad . \qquad (6)$$

2. (Value Determination Operation or VDO) Solve the system of N linear equations for all V(i) .

3. (Policy Improvement Routine or PIR) Find the worst-violated constraint in each i-group. This can be done by defining the quantity V(i,k*) as follows so that the difference V(i) - V(i,k*) is the worst (most negative) slack variable among the i-group:

$$V(i,k^*) = \underset{k \in K_i}{\text{Max}} \{r(i,k) + \beta \sum_{j=1}^{N} p(i,j,k)V(j)\} \qquad (7)$$

4. (Continuation/Termination) For each i , if $V(i,k^*)>V(i)$, write the ith initialization equation for decision $k^* \in K_i$ and go to step 2. If $V(i,k^*)=V(i)$ for every i , terminate.

Since the above procedure is identical to Howard's algorithm [55], Howard's algorithm is simply block pivoting or multiple-basis-entry solution of the dual linear program.

The system of equations represented by Equation (6), with a given decision policy, may be rewritten in matrix form where we define an N×1 column vector $V=\{V(i)\}$, and N×1 column vector $r=\{r(i,k)\}$ for a given $k \in K_i$ at each i , $P=\{p(i,j,k)\}$ as the N×N matrix of Markov transition probabilities for a given $k \in K_i$ at each i , and the N×N

identity matrix I. Equation (6) for a given policy becomes $V = r+\beta V$, or $(I-\beta P)V = r$, having the solution $V = (I-\beta P)^{-1}r$. We will call Equation (6) the VDO equation, since this is the equation solved in Howard's Value Determination Operation.

Relaxation Methods

One approach for improving Howard's algorithm would be to use relaxation methods of numerical analysis to solve the VDO equation. One such method is that of Jacobi [15], which when applied to Equation (6) involves first "solving" for $V(i)$ in the ith equation:

$$V(i) = \frac{1}{1-\beta p(i,i)}\left(r(i) + \beta \sum_{j\neq i} p(i,j)V(j)\right) \quad , \; i=1,2,\ldots,N \qquad (8)$$

(The k subscripts have been suppressed since the equations are solved for a given decision policy.)

If a starting vector of $V(i)$ values, say V^0, is used on the right, then Equation (8) for each i yields a vector V^1, which in turn used on the right leads to a vector V^2, etc., until the difference between V^n and V^{n-1} is as small as desired, at which point the approximate solution is $V=V^n$. The classical sufficient condition for convergence of Jacobi's method, when applied to Equation (8) gives

$$\sum_{j\neq i} |-\beta p(i,j)| < |1-\beta p(i,i)| \quad ,$$

which leads to $\beta<1$, so that Jacobi's method will always converge for the MDP equations treated in this thesis.

Another numerical analysis method is that of Gauss-Seidel, which is similar in application to Equation (8) to Jacobi's method, except that as soon as a new value for the ith component of V^n is found, the new value is used to compute further elements. Convergence is faster in general than that for Jacobi's method, because, on the average, new values are used sooner. The classical sufficient condition for the Gauss-Seidel procedure [15], when applied to Equation (8), gives

$$\sum_{j \neq i} \left| \frac{-p(i,j)}{1-p(i,i)} \right| \leq \mu < 1 \quad , \text{where} \quad \mu = |1-\beta| \quad ,$$

leading to $\beta<1$, so that the Gauss-Seidel procedure will always converge for problems treated here.

Not only can the methods of Jacobi and of Gauss-Seidel be used on the VDO equations (Equation (8), which is a minor modification of Equation (6)), but also on the PIR equations (Equation (7)), because Equations (6) and (8) are identical except for the maximization operator. The convergence conditions are the same, namely $\beta<1$, and there is nothing in the convergence properties of either procedure that would rule out mixing value iteration and policy iteration in any order whatsoever. Indeed, if one were to program Howard's algorithm in its original form, with either Jacobi's or Gauss-Seidel iterations substituted for direct matrix-inversion solution of each VDO, the result would consist of a mixed series of iterations: many value iterations, one policy iteration, many value iterations, one policy iteration . . . . As it turns out, no one has found a problem or class of problems for

which this particular order of mixed iterations is particularly efficient, and hence there has been a proliferation of mixed iterative methods that are distinguished from each other only in the number and order of policy iterations and value iterations.

Overrelaxation Methods

In the same spirit as the relaxation methods, a further refinement may be obtained using overrelaxation methods of numerical analysis. For example, the Accelerated Jacobi method is given by

$$V^{n+1}(i) = \omega_i\left(r(i) + \beta\sum_{j=1}^{N} p(i,j)V(j)\right) + (1-\omega_i)V^n(i) \quad , \omega_i \geq 1$$

where the $\omega(i)$'s are to be chosen experimentally. Also, if we accelerate every time a new $V(i)$ is computed in the method of Gauss-Seidel, we obtain the Accelerated Gauss-Seidel method, and if we wait until all the $V(i)$'s have been evaluated, we get the Semi-Accelerated Gauss-Seidel method. The drawback of these techniques is that there is no way other than trial and error to obtain the acceleration factors. Kushner and Kleinman [74] give computational results for the undiscounted case, and show that the techniques converge. For the discounted case, the proofs of convergence are trivial since $\beta<1$.

Mixed Iterative Procedures

Let us define policy convergence as obtaining the same decision policy in two successive policy iterations, with or without intervening value iterations. That is to say, when $k^* \varepsilon K_i$ does not change for any i in an iteration of Equation (7), policy convergence is said to have been achieved. Similarly, we define value convergence as obtaining,

for every i, a difference between consecutive values of $V(i)$ that is as small as a given difference δ, regardless of whether either or both of the consecutive iterations are policy iterations or value iterations. That is to say, when $V(i)$ or $V(i,k^*)$ does not change more than δ for any i in an iteration of either Equation (8) or Equation (7), value convergence is said to have been achieved. Finally, we define δ-optimality as achievement of both policy convergence and value convergence. That is to say, if all of the variables $k \in K_i$ remain constant and all of the variables $V(i)$ remain almost constant, δ-optimality is said to have been achieved. All the mixed iterative procedures terminate upon achievement of δ-optimality. For suitably small δ, the policy obtained is either the optimal policy or a policy so good that further possible improvement of the objective function is of the order of $\delta/(1-\beta)$, as is developed below from Equation (10).

The mixed iterative procedures to be described below can be motivated by the experimental fact that, especially in large-scale MDP problems, the computational load associated with either value iteration or direct matrix-inversion solution of VDO's is generally much higher than the computational load associated with policy iteration.

Hastings's Method. In Howard's algorithm, each PIR is one policy iteration by Jacobi's method. Hastings [49] reported a modification in which each PIR is one policy iteration by the Gauss-Seidel procedure. Savings in computational time are achieved whenever the problem is large enough so that the faster increase in some of the $V(i)$'s during each PIR allows earlier policy convergence. Thus the total number of iterations may be reduced, so that the total number of VDO's is reduced.

Successive Approximations. White [120] reported savings in computational load for large-scale problems by using a method suggested but little exploited in Howard's original work. This method has come to be known as successive approximations. VDO's are avoided, and the method simply consists of a sequence of policy iterations until δ-optimality is achieved. Of course, in those iterations for which k^* is not changed in Equation (7), and $V(i)$'s are increased; each such iteration is in fact a value iteration by Jacobi's method, except that the slight improvement of Equation (8) over Equation (6) is not realized.

White's method is especially important because of a neat correspondence with finite-horizon dynamic programming. Starting with $V(i)=0$ for all i, and using Jacobi iterations throughout, the nth vector V^n is the same as the value of the process when exactly n transitions remain and the salvage values are zero. This is obvious, since $V^0=0$, $V^1=r$, $V^2=r+\beta V^1$, etc. Thus White's method yields the finite-horizon solution for every number of transitions up to the last transition considered.

As Morton [89] observes, Bellman's suggested "value-iteration technique" [7], published in 1957, three years before Howard's method, consists precisely of finding an initial reward-function approximation in an unspecified manner, and then utilizing iterative machinery identical to what Howard later calls in a different context "policy improvement routine." This essentially is Howard's method.

Hastings' modification consists of substituting Gauss-Seidel iterations for Jacobi iterations; this modification is efficacious, as is shown later in this chapter. It also costs nothing extra in storage,

computation time or programming effort, but it does ruin the finite-horizon correspondence. A second modification, possible for many particular applications, is to start with an accurate guess as to what the optimal vector V will be, and to use this guess as the vector V^0. If the guess is very accurate, δ-optimality can often be achieved in very few iterations, because of the well-known fact that the finite-horizon dynamic programming equation for $V(i)$ approaches Equation (6) as the vector $\{V(j)\}$ on the right approaches $\{V(i)\}$ on the left.

Action-Space Reducing Techniques

MacQueen [81] gives a test for suboptimal actions in a Markov decision process, and Totten [116] reports a systematic procedure of using MacQueen's test at every stage of an iterative solution to establish bounds on the $V(i)$ values and to eliminate more and more suboptimal actions as the solution proceeds. The effect is to reduce the number of computations in each policy iteration, leaving value iterations unchanged. For large-scale problems (those with more than 100 states), the reduction in computational load would be minor even if policy iterations were done for free. For this reason, the entire class of methods that serve only to reduce the action space is of little interest here.

A Taxonomy of Micro Solution Procedures

The close relationships among the methods discussed in this chapter are emphasized by the fact that each method can be described in terms of the six blocks in Figure 5, which constitutes a taxonomy for them. Since each method is discussed in detail below, only their non-trivial taxonomic relationships will be discussed here.

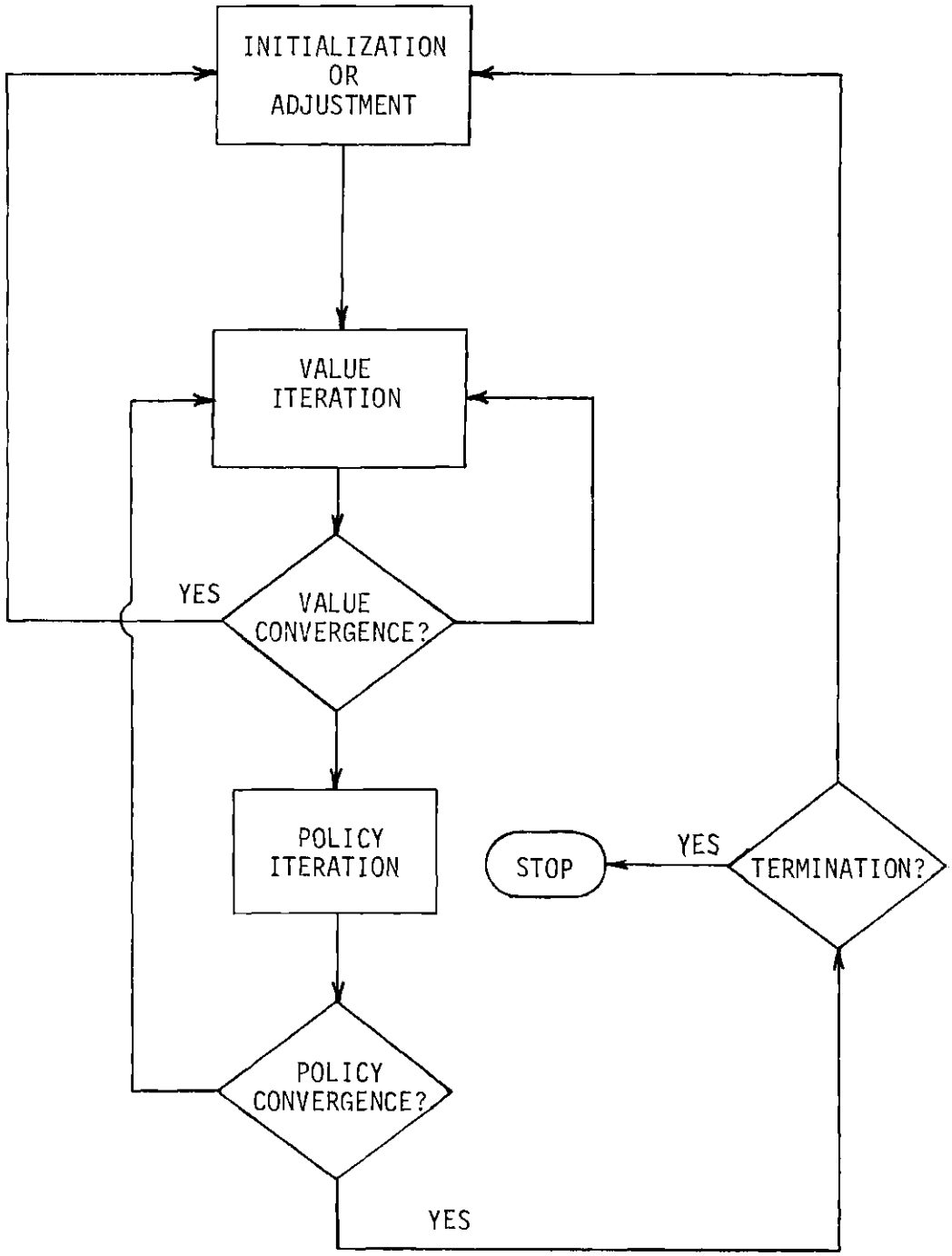


Figure 5. General Flow Chart for All Micro Methods

The initialization or adjustment block receives a policy and the values $V=\{V(i)\}$ and transmits them to the value iteration block. A more detailed description of initialization options is given in Chapter VI.

The value iteration block executes one value iteration, with the single exception of Howard's algorithm, in which simultaneous solution of Equation (6) is substituted for the operations of this block and the following convergence block.

The value convergence block following the value iteration block is null for Howard's algorithm. For all other methods, it transfers control back to the value iteration block when value convergence is not indicated or to the policy iteration block if value convergence is indicated.

The policy iteration block executes one policy iteration.

The policy convergence block transfers control to the termination block if the policy iteration does not change the policy; otherwise it transfers control to the value iteration block. If MacQueen's refinement is used, this block also makes computations to remove decisions known to be suboptimal from further consideration.

The termination block terminates the solution and transfers information elsewhere if this is a subproblem.

Example Applications

To illustrate the computational details of the solution procedures described above, a computer program "MDP" was prepared and was used to solve Beckman's "Machine Repair" problem for $\beta=0.9$ (see problem statement in Chapter II) by several variations of method. A

listing of this program and an example run is included in Appendix I. Computational results are given in the following tables. The options of the program are as follows:

Option 1: Howard's original method

Option 2: Howard's method with Gauss-Seidel policy iterations substituted for Jacobi policy iterations (Hastings' modification)

Option 3: White's method

Option 4: White's method with Gauss-Seidel policy iterations substituted for Jacobi policy iterations

Option 5: Option 2 with 4 out of every 5 VDO's suppressed

Option 6: Option 2 with 1 out of every 2 VDO's suppressed

Table 3. Sample Results
with Options 1 and 2 of MDP

	Option 1: Howard's Method		Option 2: Hastings' Modification		
n	V(1),k	V(2),k	V(1),k	V(2),k	
0	2	2	2	2	INITIALIZATION
1	27.43,2	53.52,2	27.43,2	53.52,2	VDO
2	29.93,1	53.52,2	29.93,1	53.74,2	PIR
3	34.84,1	57.03,2	34.84,1	57.03,2	VDO
4	34.84,1	57.03,2	34.84,1	57.03,2	PIR

Table 4. Sample Results
with Options 3 and 4 of MDP

	Option 3: White's Method			Option 4: Hastings' Modification		
n	V(1),k	V(2),k	n	V(1),k	V(2),k	
0	0	0	0	0	0	INITIALIZATION
1	-6.5 ,1	9.25 ,1	1	-6.5 ,1	7.79 ,1	PIR
2	-5.26 ,1	14.61 ,2	2	-5.92 ,1	13.48 ,2	PIR
3	-2.29 ,1	19.06 ,2	3	-3.10 ,1	18.33 ,2	PIR
4	1.04 ,1	22.93 ,2	4	.36 ,1	22.58 ,2	PIR
5	4.29 ,1	26.37 ,2	5	3.82 ,1	26.33 ,2	PIR
6	7.30 ,1	29.44 ,2	6	7.07 ,1	29.67 ,2	PIR
7	10.03 ,1	32.21 ,2	7	10.03 ,1	32.64 ,2	PIR
8	12.51 ,1	34.96 ,2	8	12.70 ,1	35.28 ,2	PIR
9	14.74 ,1	36.92 ,2	9	15.09 ,1	37.63 ,2	PIR
10	16.75 ,1	38.94 ,2	10	17.22 ,1	39.73 ,2	PIR
⋮			⋮			
36	33.675,1	55.862,2	35	33.839,1	56.044,2	PIR
37	33.792,1	55.979,2	36	33.947,1	56.151,2	PIR
38	33.897,1	56.084,2	37	34.044,1	56.247,2	PIR
⋮			⋮			
∞	34.84 ,1	57.03 ,2	∞	34.84 ,1	57.03 ,2	PIR

Table 5. Sample Results
with Options 5 and 6 of MDP

Option 5:
Variation of Hastings' Modification
of White's Method

n	V(1),k	V(2),k	
0	0	0	INITIALIZATION
1	-6.5 ,1	7.79,1	PIR
2	-5.92,1	13.47,2	PIR
3	-3.10,1	18.33,2	PIR
4	.36,1	22.58,2	PIR
5	3.82,1	26.34,2	PIR
6	34.84,1	57.03,2	VDO
7	34.84,1	57.03,2	PIR

Option 6:
Variation of Hastings' Modification
of Howard's Method

n	V(1),k	V(2),k	
0	0	0	INITIALIZATION
1	-6.5 ,1	7.79,1	PIR
2	-5.92,1	13.47,2	PIR
3	34.84,1	57.03,2	VDO
4	34.84,1	57.03,2	PIR
5	34.84,1	57.03,2	PIR
6	34.84,1	57.03,2	VDO

Forecast-Accelerated Methods

Two new methods will be presented in this section. In all of the mixed iterative procedures discussed in this chapter, value convergence consumes the major portion of computation time. Some experimental facts concerning value iterations, observed by this and other authors [128,53,55], are as follows:

1. Values in successive iterations tend to follow a curve of the form $f(t)=a+b\alpha^t$, where a is the eventual value, $a+b$ is the current value, α is a constant, t counts the iterations, and $f(t)$ is the value at iteration t, $t=0,1,\ldots$. That is, the plot of the $V(i)$'s has one of the following shapes:

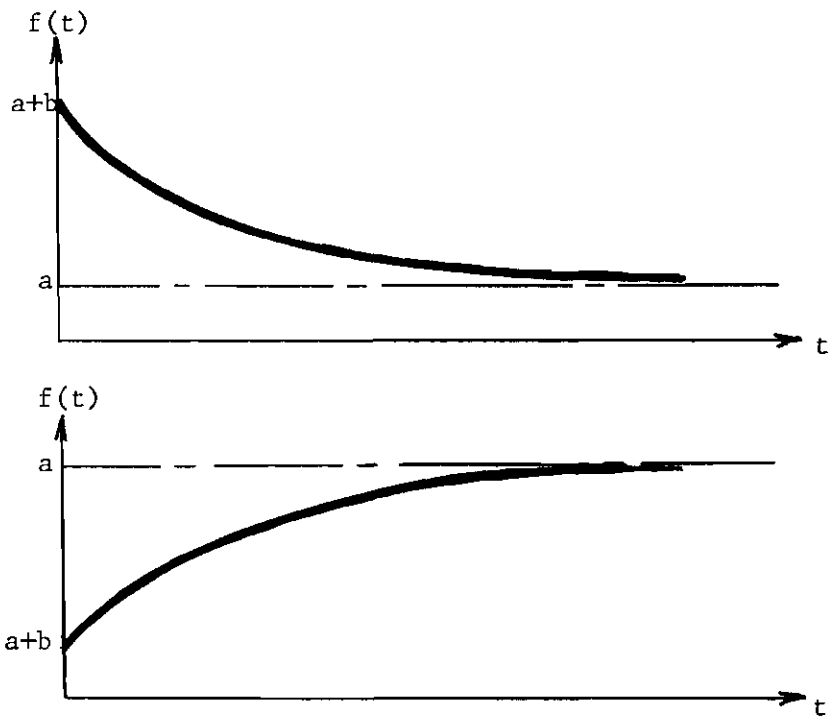


Table 6. Improvement Ratios

n	RATIO (1)	RATIO (2)
1	-.1865	.1622
2	.0299	.1121
3	.0740	.1049
4	.0899	.1020
5	.0960	.1008
6	.0984	.1003
7	.0994	.1001
8	.0997	.1000
9	.0999	.1000
10	.0999	.1000
⋮	⋮	⋮
30	.0999	.1000
31	.0999	.1000
32	.0999	.0999
33	.0999	.0999
34	.1000	.1000
35	.1000	.1000
⋮	⋮	⋮

2. The improvement ratios $\frac{V^n(i) - V^{n-1}(i)}{V^{\infty}(i) - V^{n-1}(i)}$ are constant after the effect of the initial conditions has worn off. Table 5 gives some results for the values obtained using White's method to solve Beckman's "Machine Repair" problem, presented in Chapter II, for $\beta=0.9$. Ratio(i) denotes the improvement ratio for state i .
3. Great accelerations in convergence are possible if the $V(i)$ values can be estimated. For example, in using White's successive approximation procedure on the Beckman example, where the optimal solution has $V(1)=34.84$ and $V(2)=57.03$, a "wild guess" of $V(1)=V(2)=20$ was able to reduce the number of iterations from 38 to 33.
4. Convergence is a decreasing function of β .
5. The arithmetic difference of $V(i)-V(j)$ tends to a constant as β increases to one. Also, for a given policy, $V(i)$ increases in inverse proportion to $1-\beta$; that is,

$$V(i,\beta_2) = V(i,\beta_1)\frac{1-\beta_1}{1-\beta_2}$$

From these observations, Young [128] has suggested the following procedure: the MDP problem is solved at a small value of β , so that convergence is rapid. Then, the best $V(i)$, that is, the one that seems closest to its value with the actual β , is ratioed up by the factor $(1-\beta_{small})/(1-\beta_{actual})$ to obtain a rough estimate of that $V(i)$; the other $V(i)$'s are modified so as to preserve the arithmetic

difference among the V(i) 's (otherwise, the procedure will adjust itself by changing all the V(i) 's, including the one just estimated, wasting not only iterations but also the savings to be gained when guessing at one V(i) . Then, the problem is solved at the actual β starting with the estimated V(i) 's. For example, in using Hastings' modification of White's successive approximation procedure on the Beckman example, convergence at $\beta=0.5$ was achieved in 7 iterations, yielding V(i) estimates that allowed convergence at the true $\beta=0.9$ in 7 additional iterations; the net savings was 13 iterations. This is illustrated in Table 9 which shows the output from Option 9 of the program MDP.

Two new methods motivated by the experimental facts cited above will now be introduced. They seem to be both theoretically and computationally superior to all reported acceleration procedures. The new methods are based on forecasts using two and three successive values of V(i) , respectively, and will thus be called the 2-V and 3-V methods.

Let S be an ergodic class of states. We define

$$g^{(n)} = \sum_{j \in S} p^{(n)}(i,j) r(j) \qquad (9)$$

where $p^{(n)}(i,j)$ is the probability of state j after n transitions from state i , so that $g^{(n)}$ is the expected reward to be collected on the nth transition under a given policy when the starting state at n=0 is i . Note that for large n , $g^{(n)}$ will approach g , the familiar long-run expected gain, which is defined as

$$g = \sum_{j \in S} \pi(j) r(j) \quad ,$$

where $\pi(j)$ is the long-run probability of state j, $j \in S$.

Consider two successive value iterations of $V(i)$, namely $V^n(i)$ and $V^{n+1}(i)$. Let $V^\infty(i)$ be the value to which the iterations will converge, and let $\hat{V}^\infty(i)$ be a forecast of this value. Let the iterations be done by White's method [120], so that if the policy does not change,

$$V^{n+1}(i) = r(i) + \beta \sum_{j \in S} p(i,j) V^n(j) \quad .$$

The successive values are

n	$V^n(i)$
0	0
1	$r(i)$
2	$r(i) + \beta \Sigma p^{(1)}(i,j) r(j)$
3	$r(i) + \beta \Sigma p^{(1)}(i,j) r(j) + \beta^2 \Sigma p^{(2)}(i,j) r(j)$
.	.
.	.
.	.
n	$r(i) + \beta \Sigma p^{(1)}(i,j) r(j) + \ldots + \beta^{n-1} \Sigma p^{(n-1)}(i,j) r(j)$
.	.
.	.
.	.

For any pair of successive iterations, the difference in the two values is

$$V^{n+1}(i) - V^n(i) = \beta^n \sum_{j \in S} p^{(n)}(i,j) r(j)$$

$$= \beta^n g^{(n)} \quad .$$

The difference between the eventual value $V^{\infty}(i)$ and the current value $V^n(i)$ at any iteration n is given by the terms in the series beyond n :

$$V^{\infty}(i) - V^n(i) = \beta^n \sum_{j \in S} p^{(n)}(i,j) r(j) + \beta^{n+1} \sum_{j \in S} p^{(n+1)}(i,j) r(j) + \cdots$$

Let n_0 be large enough so that for all $n \geq n_0$

$$p^{(n)}(i,j) \simeq \pi(j) \quad , \; i,j \in S \quad ,$$

so that $g^{(n)}$ for $n \geq n_0$ is approximated by the long-run expected gain per transition for the ergodic class of states S :

$$g^{(n)} = \sum_{j \in S} p^{(n)}(i,j) r(j) \simeq \sum_{j \in S} \pi(j) r(j) = g \quad .$$

With these results it follows that the ratio of the improvement in one iteration to the total improvement tends to a constant for large n :

$$\frac{V^{n+1}(i) - V^n(i)}{V^{\infty}(i) - V^n(i)} \simeq \frac{\beta^n g}{\beta^n g (1+\beta+\beta^2+\cdots)}$$

$$\simeq 1-\beta \quad . \qquad (10)$$

Equation (10) provides the basis for the δ-convergence stopping rule used in this chapter, and also for two forecast-acceleration methods.

The δ-convergence stopping rule follows from noting that if $|V^{n+1}(i)-V^{n}(i)| = \delta$, then from Equation (10), $|V^{\infty}(i)-V^{n}(i)| \simeq \delta/(1-\beta)$. Thus if we wish to stop value iterations when the difference between the current value and the true value is approximately $\delta/(1-\beta)$, we stop when the difference between the values calculated in two successive iterations is δ or less. In the examples in this chapter, with $\beta=0.9$, the stopping criterion $\delta=0.1$ always gave values within 1.0 of the true value, providing experimental evidence that Equation (10) is robust when applied not only to White's method but to other methods as well.

Equation (10) makes a "strong statement" that each iteration closes a proportion $(1-\beta)$ of the gap, and a "weak statement" that successive iterations close equal proportions of the gap. The weak statement is a generally occurring phenomenon in numerical analysis, and would be expected to occur in a wide range of iterative techniques. The strong statement, on the other hand, is specific to MDP and specific to White's unmodified method within MDP (although experimental results suggest it holds approximately for all the value iteration methods of this chapter).

3-V Method

Consider three successive value iterations with no intervening policy iterations, yielding for state i , at a given decision policy,

the values $V^{n-2}(i)=a_0$, $V^{n-1}(i)=a_1$ and $V^{n}(i)=a_2$. The unknown value $V^{\infty}(i)$ to which further value iterations will converge will be forecast; call the forecast $\hat{V}^{\infty}(i)$. The basis for the forecast is that of geometric decrease of the residual $V^{\infty}(i)-a_n$, so that the forecast value is that value for which each iteration reduces the residual by a constant proportion. This proportion is then equal to $(a_1-a_0)/(\hat{V}^{\infty}(i)-a_0)$ and also equal to $(a_2-a_1)/(\hat{V}^{\infty}(i)-a_1)$, leading immediately to the forecast

$$\hat{V}^{\infty}(i) = \frac{a_1(a_1-a_0) - a_0(a_2-a_1)}{(a_1-a_0) - (a_2-a_1)}$$

When for some i it is observed that $V^{n}(i)$ is clearly following the assumed geometric behavior, the convergence is accelerated by letting $V(i)$ jump to its forecast $\hat{V}^{\infty}(i)$. The assumed behavior is indicated by two consecutive estimates for which $|a_1-a_0|>|a_2-a_1|$ and whose sample coefficient of variation is small. When jumps were made to the latest estimate that was within 15 percent of the previous estimate, solution of the Beckman problem was accelerated by 6 iterations when the solution method was that of White and by 5 iterations when the solution method was that of Hastings' modification of White's method.

The added storage requirements for extrapolative accelerations are approximately UN added words, where U is the number of parameters used in forecasting each $V(i)$ series, and where N is the number of states. The acceleration just described adds about $3N$ words: two earlier $V(i)$ values and one forecast. If the two-β method des-

cribed earlier is viewed in this light, the added storage is negligible, since only already-stored values are used: $V(i)$, β_{small} and β_{actual} . Any extrapolative procedure that has modest storage requirements and that significantly accelerates value convergence should be advantageous for large-scale MDP.

The 3-V forecast-acceleration technique just described is similar to an empirical technique reported by Zaldivar and Hodgson [132] for use in undiscounted MDP problems, where the convergence behavior is somewhat different. They did not present a theoretical justification, but the $V(i)$ behavior in undiscounted problems is roughly that of geometric approach to a straight line with constant slope. Zaldivar and Hodgson tried forecasting g (as defined above, which is the quantity output by value iterations in undiscounted problems) both as the slope of a straight line through the last three values and also as the slope to which a negative-exponential curve through the last three values was tending toward. They reported the linear forecast as best.

2-V Method

Whereas the 3-V forecasting method described above is based on the expectation that successive iterations should close equal proportions of the gap, its development makes no use of the further expectation that these proportions are not only equal to each other but also equal to $1-\beta$. We can write Equation (10) as

$$\frac{V^n(i) - V^{n-1}(i)}{V^{\infty}(i) - V^{n-1}(i)} \simeq 1 - \beta \quad ,$$

so that if $\hat{V}^{\infty}(i)$ is a forecast of the true $V^{\infty}(i)$, we immediately have the basis for a 2-V forecasting method:

$$\hat{V}^{\infty}(i) = \frac{V^{n}(i) - V^{n-1}(i)}{1 - \beta} + V^{n-1}(i) \quad . \tag{11}$$

Convergence is indicated by two successive forecasts that have a small sample coefficient of variation, as was the case for 3-V forecasts.

The 2-V forecasts were used in solution of Beckman's problem by White's method, under the rule that both $V(1)$ and $V(2)$ were jumped to their forecast value when, for both states, the difference between two successive forecasts was less than 15 percent of the difference between two successive values of $V(i)$. The results are shown in Table 9. A jump was made after iteration 5, and δ-convergence for $\delta=0.1$ was achieved in a total of 9 iterations.

2-V forecasting as outlined above requires $2N$ extra storage words: N words for the old $V(i)$'s and N words for the old forecasts.

For all three of the new convergence aids introduced in this chapter--3-V forecast-acceleration, the two-β method, and 2-V forecast-acceleration--interesting issues arise as to how to apply them to best advantage. For example, how tight a criterion should be applied to decide whether to jump to a forecast? Should a jump be delayed until all jumps in an ergodic class of states can be made together, or should jumps be made separately? How does one choose a suitable β_{small}, or a suitable series of them, in the two-β method? The answers to these questions seem to be highly dependent on the data of each individual

problem. Further research is needed on these points before any one of these three methods can be incorporated into a large-scale general solution scheme.

Example Applications

To illustrate the use of the forecast-accelerated methods presented in the past section, the program "MDP" used with the previous six options was extended to include the following options:

Option 7: Forecast-Accelerated White's method (3-V)

Option 8: Improved Forecast-Accelerated White's method (2-V)

Option 9: Two-β method using Hastings' modification of White's technique

The results for Beckman's problem are given in the following tables.

Table 7. Sample Results
with Option 7 of MDP

Option 7:
Forecast-Accelerated
White's Method

n	V(1),k	$\hat{V}(1)$	V(2),k	$\hat{V}(2)$	
0	0		0		INITIALIZATION
1	-6.5 ,1		9.25,1		PIR
2	-5.26,1	-5.46	14.61,2	21.98	PIR
3	-2.29,1		19.06,2	40.91	PIR
4	1.04,1		22.93,2	48.85	PIR
5	4.29,1	116.82	26.37,2	53.50	PIR
6	7.30,1	45.29	29.44,2	55.65	PIR
7	10.03,1	37.83	32.21,2	56.52	PIR
8	12.50,1	35.84	34.69,2	56.84	PIR
	35.84		56.84		JUMP
9	35.20,1	35.21	56.97,2	56.97	PIR
10	34.98,1	34.85	57.01,2	57.04	PIR
11	34.90,1	34.90	57.03,2	57.04	PIR
	34.85		57.04		JUMP
12	34.85,1	34.79	57.04,2		PIR

Table 8. Sample Results
with Option 8 of MDP

Option 8:
Improved Forecast-Accelerated
White's Method

n	V(1),k	$\hat{V}(1)$	V(2),k	$\hat{V}(2)$	
0	0		0		INITIALIZATION
1	-6.5 ,1	-64.99	9.25,1	92.5	PIR
2	-5.26,1	5.87	14.61,2	62.84	PIR
3	-2.29,1	24.41	19.06,2	59.12	PIR
4	1.04,1	31.09	22.93,2	57.78	PIR
5	4.28,1	33.49	26.37,2	57.30	PIR
	33.49		57.30		JUMP
6	34.36,1	42.14	57.13,2	55.57	PIR
7	34.67,1	37.47	57.06,2	56.51	PIR
8	34.78,1	35.79	57.04,2	56.84	PIR
9	34.82,1	35.18	57.04,2	56.96	PIR

Table 9. Sample Results
with Option 9 of MDP

Option 9:
Two-β Method Using Hastings'
Modification of White's Technique

n	β	V(1),k	V(2),k	
0	.5	0	0	INITIALIZATION
1		-6.5 ,1	8.44,1	PIR
2		-6.02,1	11.66,1	PIR
3		-5.09,1	12.99,1	PIR
4		-4.53,1	13.55,1	PIR
5		-4.24,1	13.80,1	PIR
6		-4.11,1	13.91,1	PIR
7		-4.05,1	13.96,1	PIR
		Updating β:		
	.9			INITIALIZATION
		69.8	51.78	PIR
8		48.21,1	55.05,1	PIR
9		39.97,1	55.89,2	PIR
10		36.63,1	56.27,2	PIR
11		35.30,1	56.45,2	PIR
12		34.79,1	56.56,2	PIR
13		34.61,1	56.63,2	PIR
14		34.55,1	56.68,2	PIR

CHAPTER V

LARGE SCALE MDP PROBLEMS

The study of large-scale programming techniques for the solution of MDP problems is one of major importance. However, from the large body of scholars that have studied MDP, only a handful of references in this direction can be found. This is not to say that practitioners of MDP are not aware of this situation. As Miller [86] states,

> A well-known limitation in applying dynamic programming as a computational technique is that many "real world" sequential decision models must be described by an extremely large number of states. For such sequential decision models there are generally two approaches open to the analyst: he can develop a heuristic dynamic (state-dependent) decision rule, or, he can recast the problem into a static framework and apply a variant of mathematical programming to obtain a static (non-state-dependent) decision rule. Typically this ends the analysis except for perhaps some simulation tests to evaluate the proposed procedures.

A large-scale MDP problem will be defined here as one with a large state space. It is true that there are some problems with large action spaces which may be considered by some authors as belonging in this category. However, from the solution methods of Chapter IV, it is clear that a large action space simply implies that there will be many actions to be evaluated for some states at each PIR. However, this is not a serious drawback, since the amount of computations required to evaluate a given action is negligible. Additionally, tests have been developed to eliminate from the analysis suboptimal actions [108,116]. On the other hand, a large state space represents not only more states

with their corresponding actions to be evaluated at each PIR, but more important than that, larger systems of equations to be solved at each VDO. The computational and storage requirements of a VDO are the critical issues in the solution of large-scale MDP problems. Therefore, the procedures proposed in Chapter VI will deal with the solution of problems with a large state space.

A Classification of Solution Methods for Large-Scale MDP Problems

To put in a proper perspective the results reported in this thesis, we may consider the following solution strategies for large-scale MDP problems. The first class of strategies deals with reformulation alternatives, while the second class treats solution methods.

A) Work with a Reduced Problem

1. Recast the problem into a deterministic dynamic problem, and solve this problem to obtain an approximate solution. Norman and White [92] used this approach to initialize their approximate method of expectations. The limitations of this strategy are obvious, and it will only yield satisfactory results for those problems which should have been formulated as deterministic in the first place.

2. Reformulate the dynamic stochastic problem into a static framework, and solve the reduced state-independent stochastic problem. This has been the most popular method of solution. However, it is clear that this strategy will be dominated by one that uses all avail-

able information. Besides, it seems wasteful to formulate correctly a problem and then adopt an inferior solution. An example in stochastic location analysis will be given next to illustrate the use of dynamic vs static decision rules.

3. Approximate the given problem by a smaller dynamic stochastic problem with a reduced state space. This is also a widely used procedure, and it will be illustrated with an example of a dispatching problem. The methodology presented in Chapter III for statistical inference on MDP can be used to gather the information required by the reduced problem.

B) Solve the Problem Indirectly

1. Use additional information on the form of the optimal solution to restrict the search to a given class of policies, called heuristic decision rules or structured policies by some authors [86,96]. The main object of such studies is to find sufficient conditions for the optimality of structured policies. This strategy has been very successful in the fields of stochastic inventory models [118], replacement models [26], and in some other special processes. A recent study by Porteus [96] gave some new results on the optimality of structured policies that link the studies of Strauch [114], Blackwell [12,13], and others, where general policies for

general processes are considered, with studies such as those by Scarf [107] and Derman [26], where structured policies for special processes are considered. One of the main advantages of the state-change formulation presented in Chapter II is that separating the choice moves from the chance moves allows for the identification of structured policies. For example, state-change formulation of certain inventory problems leads directly to the simple policy structure known as "S,s" policies.

C) Solve the Problem Directly

1. Use directly one of the micro methods of Chapter IV. This has been a common alternative, since for a long time no other procedures were developed. However, these methods reach storage and computation limits on any computer for problems whose size still falls within reasonable information-gathering limits.
2. Use a decomposition scheme. This seems to be the most natural way of solving large problems.

Some Examples of Large-Scale Problems

The accurate formulation of many problems as MDP requires a large number of states. In this section we give some illustrative examples of problems reported in the literature. Out of some 600 papers on MDP, only a handful refer to solutions of real-life problems, and even less of large-scale problems. To make matters worse, with the

exception of a problem of stochastic location analysis studied by Rosenthal [99], those that correspond to large problems have been solved by the use of heuristic decision rules or as static problems.

A Stochastic Location Analysis Problem

Some problems of stochastic location analysis have been studied by Rosenthal [99]. We present a small example from his work to illustrate the distinction between a dynamic or state-dependent policy and a static or state-independent policy.

Consider a situation where two facilities, a server and a customer, change their locations weekly over an infinite horizon. Both facilities are confined to a finite set $S=\{1,2,\ldots,N\}$ of possible locations. The customer's location changes according to an $N\times N$ Markov transition matrix; the server's location depends on the instructions of a decision maker. Costs are incurred in two ways: 1) when the server is relocated, and 2) when the server and the customer interact for service. At time t, $t=0,1,2,\ldots$, the state of the system is $(X_t,A_t)\in S\times S$, where X_t is the server location and A_t is the customer location. Given X_0,A_0, a positive relocation factor α, a discount factor β, an $N\times N$ transition matrix P and an $N\times N$ distance matrix D, the process is an follows:

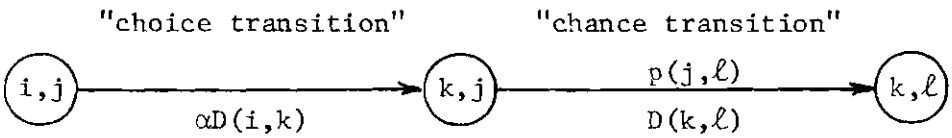
1. The decision maker observes (X_{t-1},A_{t-1}) and chooses X_t.
2. The relocation cost $\alpha D(X_{t-1},X_t)$ is incurred.
3. The customer location A_t is realized, and
4. The service cost $D(X_t,A_t)$ is incurred.

The process is repeated indefinitely. The objective is to choose the server locations so as to minimize the expected discounted sum of costs:

$$\text{Min } E \sum_{t=1}^{\infty} \beta^t \left[\alpha D(X_{t-1}, X_t) + D(X_t, A_t)\right]$$

Formulation as a State-Change Problem. Let (i,j) be a typical state. A dynamic (state-dependent) decision rule is: for each state (i,j), specify $k(i,j)$, where $k(i,j)$ is the location to which the server is moved whenever state (i,j) is observed. A static (state-independent) decision rule would specify a permanent server location, or it would choose in advance the sequence of locations $X_1, X_2, \ldots$ in advance at time 0. The advantage of dynamic decision rules, which encompass static decision rules as special cases, should be clear since they take into account up-to-date information.

Moving the server constitutes the "state-change." A typical transition with $k=k(i,j)$ is given below.



where

$\alpha D(i,k)$ = state-change cost

$D(k,\ell)$ = cost resulting from the chance-transition visit to state (k,ℓ)

$p(j,\ell)$ = transition probability of the chance-transition visit to state (k,ℓ)

If we denote as usual $V(i,j)$ the value of state (i,j), then for all states (i,j) we have the MDP problem defined by

$$V(i,j) = \min_{k}\{\alpha D(i,k) + \sum_{\ell} p(j,\ell)D(k,\ell) + \beta\sum_{\ell} p(j,\ell)V(k,\ell)\}$$

Note that for practical location analysis problems, this will be a large problem since the number of states is N^2 where N is the number of feasible locations.

Dispatching in a Recoverable-Item Inventory System

An interesting large-scale inventory problem can be defined as follows: Given N reparable items, one service center, and M customers demanding these items, find a dispatching rule which assigns the newly available inventory items coming out of repair to the customer in the system with the greatest need, so as to minimize the long-run average expected number of backorders.

The model arises from an Air Force inventory problem in which there are M bases, N items, say aircraft engines, and a repair center called the depot. Miller [86] formulated the problem as a finite state MDP and showed that a heuristic rule is optimal.

An Inventory Model

The MDP formulation of a stochastic inventory problem was first given by Bellman [7] in 1957 and then by Iglehart [56] in 1963. However, very little use of this formulation has ever been made, despite the great interest of many well known authors in the field of inventory theory. The two basic reasons for this fact are data gathering difficulties and the computational difficulties arising from the resulting large problems. The fact that a large state space will be required arises from the definition of a state as the number of units at hand in

inventory. Thus, if we consider having 1000 units at hand there will be 1001 states in the model. If we try grouping the items to constitute states, say one state being equivalent to 10 items, then the number of units ordered and demanded are restricted to multiples of the group size, and there is no satisfactory basis for grouping the probability data so as to preserve the Markov property.

The state-change formulation given in Chapter II is far superior to those previously mentioned for the inventory-control problem, in that it simplifies the information gathering process by separating choice and chance moves, as is illustrated by the inventory model treated in Chapters II and VII of this thesis. In the literature, starting with Wagner's early paper [118], emphasis has been given to the simplifying conditions under which heuristic decision rules such as the (S,s) rule and its generalizations can guarantee optimality. With the state-change formulation and the solution procedures established in this thesis, it now becomes routinely possible to treat inventory problems that do not meet these simplifying conditions.

A kth Order MDP Problem

A well-known limitation of ordinary (first-order) Markov chains is the lack of memory of the stochastic process, formalized by the Markovian assumption which simply states that the conditional probability of the process' being in some state in the next transition, given the present state, is independent of the history of the process. The kth-order MDP problem has not been reported in the MDP literature, and is included here to illustrate how a large class of problems can be handled. Additionally, it will be noted that the resulting transition

matrix has a very interesting structure which will be exploited in Chapter VI.

It is well known that all higher order Markov chains can be represented by a first-order chain with an increased state space. For example, a second order Markov chain with state space $\{1,2\}$ can be formulated as one of first order with state space $\{(1,1),(1,2),(2,1),(2,2)\}$ with the following transition matrix:

$$P = \begin{array}{c} \\ (1,1) \\ (1,2) \\ (2,1) \\ (2,2) \end{array} \begin{array}{c} \begin{array}{cccc} (1,1) & (1,2) & (2,1) & (2,2) \end{array} \\ \begin{bmatrix} p_{11} & p_{12} & 0 & 0 \\ 0 & 0 & p_{11} & p_{12} \\ p_{21} & p_{22} & 0 & 0 \\ 0 & 0 & p_{21} & p_{22} \end{bmatrix} \end{array}$$

An important point is that the transition matrix is very sparse. This idea will be followed in Chapter VI.

Reported Solution Methods for Large-Scale MDP Problems

Norman and White [92] are among many authors who point out that Howard's iterative method is not well suited to the solution of large-scale MDP problems. In this section we review the reported studies dealing with the optimization of large problems.

Method of Successive Approximations

The method of successive approximations was discussed in Chapter IV, and was described as a sequence of PIR's for solving a MDP problem. This method has been cited by Zaldivar and Hodgson [132] as being the

most widely used method for large-scale problems. The reason for its popularity is that it requires no VDO's, and the PIR's are not affected as much by a large state space, since the evaluations at a given state consist of computing the quantity

$$r(i,k) + \beta \sum_{j=1}^{N} p(i,j,k)V(j)$$

which requires storage of the transition probabilities and immediate rewards associated only with that state. For large problems, the full storage of $p(i,j,k)$ can be relegated to a backup storage unit. This is also the reason for the use of the iterative method of Gauss-Seidel in numerical analysis.

An Approximate Method of Solution for MDP Problems Using Expectations

An approximate method of solution for MDP problems using expectations is due to Norman and White [92], and belongs to the first class of solution methods for large-scale MDP problems. Their development was for the undiscounted case, but we give the corresponding discounted version. Their basic idea is to consider the visited states as random variables, where the states are assumed to have some physical meaning (as in the inventory problem, where the states correspond to the number of units on hand). A deterministic dynamic problem is then defined by replacing, for each action k at state i, the random variable j by its expected value $\mu(i,k)$ given as

$$\mu(i,k) = \sum_{j=1}^{N} j\, p(i,j,k)$$

Let the values for this reduced problem be distinguished from those of the original problem by denoting them as $V^*(i)$, $i=1,2,\ldots,N$.

They propose a two-stage approximation: First, solve the reduced deterministic problem, which can be stated as

$$V^*(i) = \max_k \{r(i,k) + \beta V^*[\mu(i,k)]\} \quad i=1,2,\ldots,N$$

To solve this problem, we can modify Howard's method as shown in Figure 6. Notice that Figure 6 is just a variation of Howard's method depicted in Figure 5, Chapter IV.

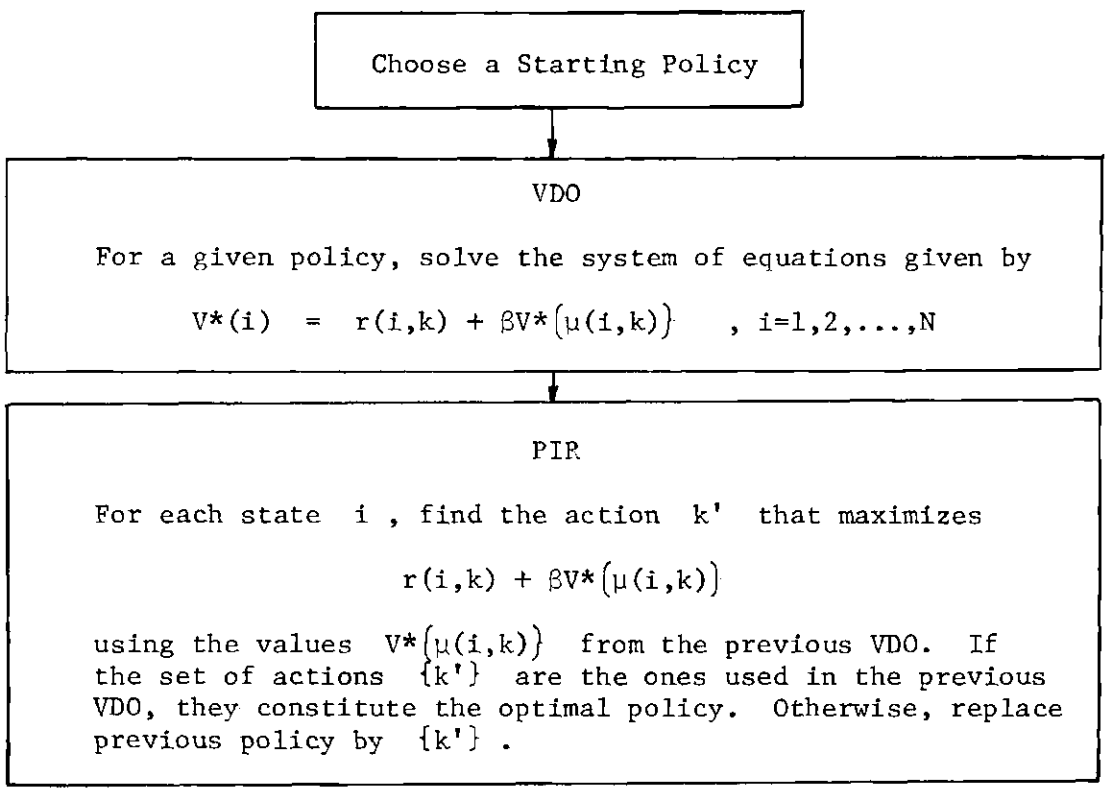


Figure 6. First Stage of
The Norman and White Solution Procedure

The next step is to perform a single PIR as in Howard's method, using the reward function obtained from the deterministic problem, to find the approximate policy. The overall solution strategy can be portrayed by the following figure:

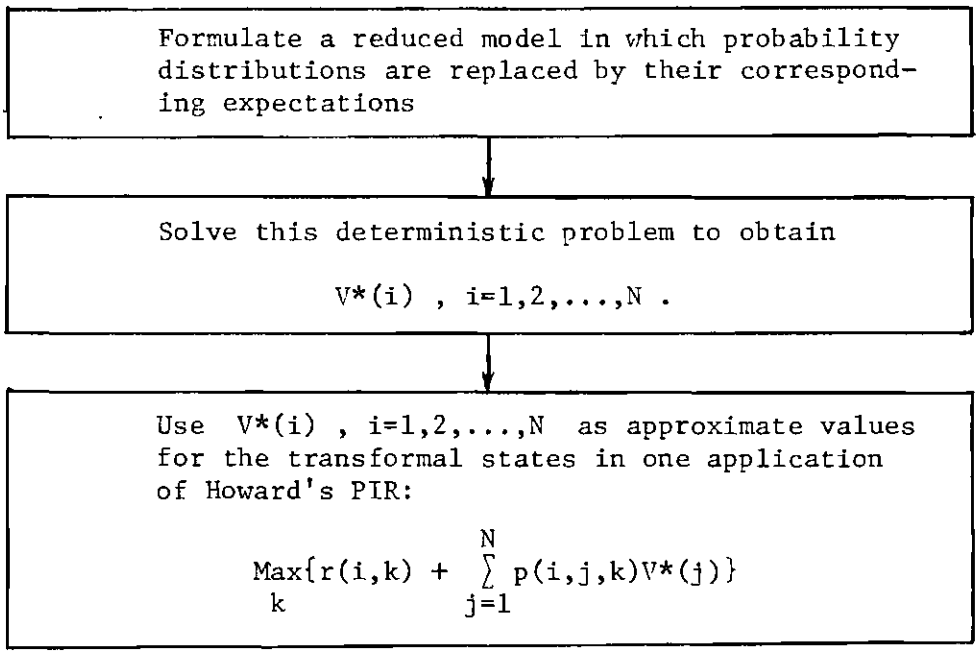


Figure 7.
The Norman and White Solution Procedure

An Example. To illustrate this method, we solve Beckmann's "Machine Care Problem" presented in Chapter II, for $\beta=0.9$.

1. Find the transformal states that can be reached under action k from state i , $i=1,2,\ldots,N$, $k=1,2,\ldots,k(i)$.

   State 1

   Action 1: $\mu(1,1) = 1\ p(1,1,1) + 2\ p(1,2,1)$

   $= 1\ (.50) + 2\ (.50) = 1.5 \simeq 1$

Action 2: $\mu(km2) = 1p(1,1,2) + 2p(1,2,2)$

$= 1(.33) + 2(.67) = 1.67 \simeq 2$

State 2

Action 1: $\mu(2,1) = 1p(2,1,1) + 2p(2,2,1)$

$= 1(.25) + 2(.75) = 1.75 \simeq 2$

Action 2: $\mu(2,2) = 1p(2,1,2) + 2p(2,2,2)$

$= 1(.10) + 2(.90) = 1.90 \simeq 2$

An interesting comparison may be established between the stochastic activity networks corresponding to the stochastic and deterministic problems, as illustrated on the next figure.

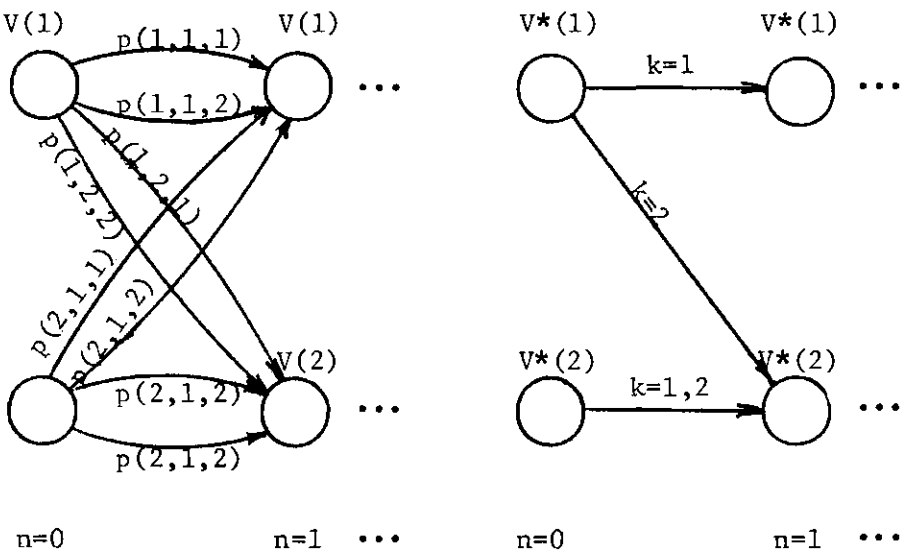


Figure 8. Comparison of Stochastic and Reduced Deterministic Problems

2. Choose a starting policy, and solve the deterministic problem. Let the starting policy be $(1,1)$. Then, we need to solve the system

$$\begin{aligned} V^*(1) &= r(1,1) + \beta V^*\{\mu(1,1)\} \\ &= -6.5 + \beta V^*(1) \end{aligned}$$

$$\begin{aligned} V^*(2) &= r(2,1) + \beta V^*\{\mu(2,1)\} \\ &= 9.25 + \beta V^*(2) \end{aligned}$$

from which

$$V^*(1) = -65$$

$$V^*(2) = 92.50$$

Next we do a PIR as follows:

State 1

$$\text{Action 1: } r(1,1) + \beta V^*\{\mu(1,1)\} = r(1,1) + \beta V^*(1) = -6.5 + .9(-65) = -65$$

$$\text{Action 2: } r(1,2) + \beta V^*\{\mu(1,2)\} = r(1,2) + \beta V^*(2) = -12.99 + .9(92.50) = 70.26$$

$\therefore k'=2$, $V^*(1) = 70.26$

State 2

$$\text{Action 1: } r(2,1) + \beta V^*\{\mu(2,1)\} = r(2,1) + \beta V^*(2) = 9.25 + .9(92.50) = 92.50$$

$$\text{Action 2: } r(2,2) + \beta V^*\{\mu(2,2)\} = r(2,2) + \beta V^*(2) = 7.70 + .9(92.50) = 90.95$$

$\therefore k'=1$, $V^*(2) = 92.50$

Since the policy $(2,1)$ was not our previous policy, we do another iteration.

Second Iteration:

VDO: $V^*(1) = r(1,2) + \beta V^*\left(\mu(1,2)\right)$

$= -12.99 + .9V^*(2)$

$V^*(2) = r(2,1) + \beta V^*\left(\mu(2,1)\right)$

$= 9.25 + .9V^*(2)$

from which

$V^*(1) = 70.26$

$V^*(2) = 92.50$

PIR:

State 1

Action 1: $r(1,1) + \beta V^*\left(\mu(1,1)\right) = r(1,1) + \beta V^*(1)$

$= -6.5 + .9(70.26) = 56.73$

Action 2: $r(1,2) + \beta V^*\left(\mu(1,2)\right) = r(1,2) + \beta V^*(2)$

$= -12.99 + .9(92.50) = 70.26$

$\therefore k'=2 \quad , \quad V^*(1) = 70.26$

State 2

Action 1: $r(2,1) + \beta V^*\left(\mu(2,1)\right) = r(2,1) + \beta V^*(2)$

$= 9.25 + .9(92.50) = 92.50$

Action 2: $r(2,2) + V^*(2,2) = r(2,2) + V^*(2)$

$= 7.70 + .9(92.50) = 90.95$

$\therefore k'=1 \quad , \quad V^*(2) = 92.50$

The policy (2,1) is the same as the previous policy, thus we have value and policy convergence, and the solution for the deterministic problem is

$V^*(1) = 70.26$

$V^*(2) = 92.50$

From Figure 8 we see that under this policy the process would be in State 2 after the first transition for any initial state.

3. Perform a PIR with the values of $V^*(1)$, $V^*(2)$.

State 1

Action 1: $r(1,1) + \beta\left[p(1,1,1)V^*(1) + p(1,2,1)V^*(2)\right]$
$= -6.5 + .9\left[.50(70.26 + .50(92.50)\right] = 34.19$

Action 2: $r(1,2) + \beta\left[p(1,1,2)V^*(1) + p(1,2,2)V^*(2)\right]$
$= -12.99 + .9\left[.33(70.26) + .67(92.50)\right] = 63.65$

$\therefore k'=2 \quad , \quad V^*(1) = 63.65$

State 2

Action 1: $r(2,1) + \beta\left[p(2,1,1)V^*(1) + p(2,2,1)V^*(2)\right]$
$= 9.25 + .9\left[.25(70.26 + .75(92.50)\right] = 87.50$

Action 2: $r(2,2) + \beta\left[p(2,1,2)V^*(1) + p(2,2,2)V^*(2)\right]$
$= 7.70 + .9\left[.10(70.26 + .90(92.50)\right] = 88.95$

$\therefore k'=2 \quad , \quad V^*(2) = 88.95$

The approximate solution to the problem is

For State 1, choose Action 2: $V^*(1) = 63.65$

For State 2, choose Action 2: $V^*(2) = 88.95$

Notice that the exact solution to this problem found in Chapter IV was:

For State 1, choose Action 1: $V^*(1) = 34.84$

For State 2, choose Action 2: $V^*(2) = 57.03$

which is different, both in policy and values, to the approximate solution.

Norman and White give one example due to Bellman [7] and one due to Howard [55], and for these problems their technique yields very good results. However, a closer look at the examples used to illustrate this technique reveals that the only reasons Norman and White obtained such good results are the fact that Howard's car replacement problem is essentially deterministic, and Bellman's inventory problem is essentially a one-period horizon problem. These facts were first pointed out by Morton [89], who additionally shows with an example due to Iglehart [56], that the possible percentage cost error of this procedure is unbounded.

Decomposition Methods

Whereas most of the procedures discussed in this chapter are intended to decrease computation time, decomposition methods usually have the effect of decreasing storage requirements at the expense of increasing computation time. Decomposition methods are those methods in which the problem is actually or effectively decomposed into a group of smaller problems, each to be solved separately. The remaining chapters of this thesis are concerned mainly with decomposition methods.

Generalized Linear Programming. If we couch either the standard formulation or the state-change formulation directly in terms of variables $d(i,k)$ or $d(i,\ell)$, where these variables represent the probability of making the given decision when state i is observed, we obtain constraints that have decision variables multiplied together. An example is Equation (6) of Chapter II, where $d(i,\ell)$ and $V(j)$ are multiplied together, destroying the direct linear programming interpretation of the problem. In the case of MDP, it is possible to obtain the

linear program from first principles, so that this is in fact not a difficulty. However, Wolfe's procedure called generalized linear programming, as reported for example by Lasdon [77], offers a completely general method for obtaining a linear program from a large class of problems that includes MDP as a fairly typical special case. (In fact, the usual textbook examples of generalized linear programming are MDP problems.)

Generalized linear programming is of interest not only because of its generality, but also because of the natural basis for decomposition that exists as a by-product of the formulation. This has been recognized, but not exploited, by Kushner and Chen [76]. They present a technique for "essentially" decomposing a control system that is equivalent to undiscounted MDP, and they give a Markov interpretation to each subsystem. Their aim in decomposing the system is to use the Dantzig-Wolfe decomposition technique for accelerated convergence in solving problems that are slightly more general than MDP. The added generality stems from their recognition of the fact that, given a linear program arising from MDP, it is possible to add linear constraints to the problem without seriously increasing the difficulty of solution using their method. (The methods in this thesis, by contrast, depend critically on the structure of the constraints, and additional constraints cannot be added without danger of losing the guarantees of convergence.) Thus the problem is treated as a linear program that has a high degree of structure from the fact that most of its constraints are those of MDP.

Because of the added constraints, the Kushner and Chen procedure

is not an exact algorithm but rather a philosophy for decomposing a class of problems. Their master problem corresponds roughly to the set of constraints for the transient states, and their subproblems correspond to the sets of constraints for each ergodic class of states (or to classes of states that are made ergodic by temporary suppression of their coupling to other states). Thus their procedure is fairly closely related to the SMDP algorithm developed in the next chapter, where it is shown that it is possible to decompose MDP problems to great computational advantage when sufficient natural structure exists.

CHAPTER VI

PROPOSED MACRO SOLUTION PROCEDURES

In the classification of solution methods for large-scale MDP problems given in Chapter V, the techniques proposed in this chapter vall into category C2--the direct solution of a problem using a decomposition technique.

There are two main reasons for decomposing any type of large-scale problem, including MDP:

1. If an exploitable natural structure is present, and if computation time grows more rapidly than linearly with problem size, then decomposition may yield savings in computation time as well as in storage.
2. If the problem exceeds the capacity of efficient storage, then decomposition may allow for a favorable trade-off of reduced storage against increased computation time.

Many MDP problems do have an exploitable natural structure, and computation time for MDP is well known to grow more rapidly than linearly with problem size. The SMDP solution method presented in this chapter is applicable to such problems.

On the other hand, many large-scale problems that may not have a natural structure that invites decomposition are simply too large to solve. The DMDP solution method presented in this chapter is applicable to such problems.

This chapter also presents a general decomposition framework for solving very large problems using a combination of the two approaches.

In general, SMDP (where "S" stands for "sparse") applies to sparse transition matrices, where there are likely to occur ergodic subclasses of states, so that natural decomposition into these subclasses is efficient. DMDP (where "D" stands for "dense") applies in general to dense transition matrices, where little natural structure is likely. In DMDP the decomposition is not natural, but arbitrary. Both SMDP and DMDP are mainly directed at helping to find values at a given policy, i.e., to perform VDO's. In the SMDP framework, the policy iterations are done on the entire problem, since the problem structure changes with a change in policy; in the DMDP framework, the policy iterations are done on the subproblems.

These distinctions between SMDP and DMDP suggest a general framework for selecting a macro solution method.

VDO for Problems with a Sparse Transition Matrix

As discussed in Chapter IV, VDO consists of solution of the system of linear equations given by $(I-\beta P)V=R$. In this section we discuss classical methods of solution for sparse systems of equations, and propose the SVDO method. The SVDO method will be shown to be more efficient than the general methods which do not take advantage of the specific structure, when there is a strong structure present.

Solution of Sparse Systems of Linear Equations. There is a large body of knowledge in the field of numerical analysis dealing with problems consisting of sparse matrices. Some of the most relevant results are contained in the works of Gustavson [44], Jacques de Ruchet [57],

Jennings and Tuff [58], Rose and Bunch [100], Walsh [119], Willoughby [125], and Young [130].

In what follows we denote the system of equations by $AX=b$ where A is an $N\times N$ matrix with elements $a(i,j)$. We assume that A contains a large number of zero elements. There are two basic approaches: the first considers the position of the non-zero $a(i,j)$'s as fixed, and the second modifies the given system by ordering the elements of the matrix A by permuting its rows and columns to obtain a more convenient form for solution.

When the non-zero elements are fixed, we may classify the solution methods as direct or iterative. The direct methods involve a fixed number of arithmetic operations while the iterative methods consist of the repetition of certain steps until the required accuracy is achieved.

In considering direct methods, it is useful to distinguish two main types of matrices, the band (and circulant band) matrix, or the general sparse matrix.

A band matrix may be defined by the conditions

$$a(i,j) = 0 \qquad \begin{array}{r} i-J\geq S \\ \text{or } j-i\geq t \\ \text{with } S,t\leq N \end{array}$$

The total band width is $k=S+t-1$, and the band is symmetrically placed about the main diagonal if $S=t$. An example with $S=2$, $t=3$, $N=7$ is

$$A = \begin{bmatrix} X & X & X & & & & \\ X & X & X & X & & & \\ & X & X & X & X & & \\ & & X & X & X & X & \\ & & & X & X & X & X \\ & & & & X & X & X \\ & & & & & X & X \end{bmatrix}$$

where the symbol X denotes elements which may be non-zero. The band width here is $k=2+3-1=4$.

A circulant band matrix is a band matrix with some additional non-zero components in some or all positions (i,j) for which

$$N + i - J < S$$

$$\text{or} \quad N + J - i < t$$

thus, for the previous values of N , S , and t we have

$$A = \begin{bmatrix} X & X & X & & & & X \\ X & X & X & X & & & \\ & X & X & X & X & & \\ & & X & X & X & X & \\ & & & X & X & X & X \\ X & & & & X & X & X \\ X & X & & & & X & X \end{bmatrix}$$

The circulant band can be reduced to a simple band by renumbering the variables, resulting in a wider band.

The band form is useful only when the band width k is considerably less than N ; otherwise the solution methods for a general sparse matrix should be considered.

The main methods of solution for band matrices are:

1. Gaussian elimination with interchanges
2. Triangular factorization without interchanges, which is stable for symmetric positive-define matrices and for diagonally dominant matrices.

These methods have also been used for a general sparse matrix.

The second approach to the solution of large sparse systems of equations considers the problem of ordering the states into classes by permutations of rows and columns. Rose and Bunch [100] call this the "partitioning problem," and consider the arithmetic and memory costs.

The State-Classification algorithm described in the next section belongs to this last category of solution approaches, taking advantage of the natural decomposition obtained by classifying the states into ergodic and transient classes.

State-Classification VDO Method

In this section is reported an efficient new algorithm, jointly developed by Rosenthal [99] and the present author, for reducing the VDO computation load for MDP problems having ergodic subchains--so-called "polydesmic" chains [55]. An essential part of the new algorithm is the identification of ergodic subchains and transient states, and for this part we have used the method of Fox and Landi [40].

For use in the VDQ equations, whose matrix form is $V=r+PV$, let P be put in canonical form as follows:

$$P = \begin{array}{c|cccccc} C_1 & P_1 & & & & & \\ C_2 & & P_2 & & & & \\ \vdots & & & \ddots & & & \\ C_k & & & & P_k & & \\ C_{k+1} & R_{k+1,1} & R_{k+1,2} & \cdots & R_{k+1,k} & Q_{k+1} & \\ C_{k+2} & R_{k+2,1} & R_{k+2,2} & \cdots & R_{k+2,k} & R_{k+2,k+1} & Q_{k+2} \\ \vdots & \vdots & & & \vdots & \vdots & \ddots \\ C_n & R_{n,1} & R_{n,2} & \cdots & R_{n,k} & R_{n,k+1} & \cdots \quad Q_n \end{array}$$

Here $C_1, C_2, \ldots, C_n$ are equivalence classes such that $C_1, C_2, \ldots, C_k$ are those with recurrent states, and $C_{k+1}, C_{k+2}, \ldots, C_n$ are those with transient states.

When the elements of V and r are partitioned in the same fashion, the system of VDO equations becomes

$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_k \\ V_{k+1} \\ V_{k+2} \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_k \\ r_{k+1} \\ r_{k+2} \\ \vdots \\ r_n \end{bmatrix} + \beta \times \begin{bmatrix} & & & \\ & & & \\ & & P & \\ & & & \\ & & & \end{bmatrix} \times \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_k \\ V_{k+1} \\ V_{k+2} \\ \vdots \\ V_n \end{bmatrix}$$

or,

$$V_1 = r_1 + \beta P_1 V_1$$

$$V_2 = r_2 + \beta P_2 V_2$$

$$\vdots$$

$$V_k = r_k + \beta P_k V_k$$

$$V_{k+1} = r_{k+1} + \beta\left(R_{k+1,1}V_1 + R_{k+1,2}V_2 + \cdots + R_{k+1,k}V_k\right) + \beta Q_{k+1}V_{k+1}$$

$$V_{k+2} = r_{k+2} + \beta\left(R_{k+2,1}V_1 + R_{k+2,2}V_2 + \cdots + R_{k+2,k+1}V_{k+1}\right) + \beta Q_{k+2}V_{k+2}$$

$$\vdots$$

$$V_n = r_n + \beta\left(R_{n,1}V_1 + R_{n,2}V_2 + \cdots + R_{n,n-1}V_{n-1}\right) + \beta Q_n V_n$$

With the states partitioned as shown, the VDO that solves the entire $N \times N$ system of equations can be replaced by an ordered series of n smaller VDO's: First the k independent systems $V_1, V_2, \ldots, V_k$ are each solved independently. Then successively for $i=k+1, k+2, \ldots, n$ we replace the vector r_i with a vector r_i' that contains all currently known information:

$$r_i' = r_i + \beta(R_{i,1}V_1 + R_{i,2}V_2 + \cdots + R_{i,i-1}V_{i-1})$$

This allows, again, independent solution of each system evaluated in the succession $k+1, k+2, \ldots, n$ with the previous V's known and incorporated in r_i' :

$$V_i = r_i' + \beta Q_i V_i \quad , \; i=k+1, k+2, \ldots, n$$

Each of the reduced-size VDO's may of course be performed by any of the appropriate micro methods discussed in Chapter IV.

SVDO (Sparse VDO) Computer Code. Appendix II gives a listing of the Fortran code SVDO (Sparse VDO). Except for minor modifications, it is exactly as written and used by Richard E. Rosenthal, whose authorship is gratefully acknowledged. The code incorporates four elements: (1) Fox-Landi classification of the k ergodic subchains, (2) ordering of the remaining transient states into $n-k$ classes, (3) updating of the vector r_i for each transient class to include V's currently known and (4) solution of each reduced VDO by any one of three methods. The output of SVDO is a set of values $V(i)$ for a given policy. The sobriquet "sparse" suggests that when the transition matrix is sparse it

is normally fruitful to try this natural classification rather than to use the arbitrary classification scheme of DMDP that is normally applicable to dense transition matrices.

Elements (1), (2) and (3) of SVDO are combined into a single procedure that can be explained in Fox-Landi notation as follows: Define a node-node incidence matrix B^1 with elements

$$b^1(i,j) = \begin{cases} 1 & \text{if } p(i,j)>0 \text{ and } i\neq j \\ 0 & \text{otherwise.} \end{cases}$$

Boolean logical operations are applied to B^1 to search for a set of communicating states. Rows and columns corresponding to communicating states are replaced by their respective unions (logical "or" with 1 corresponding to "true"), yielding a smaller matrix B^2. A search of B^2 similarly yields B^3, etc., until an ergodic chain is identified by having a set of communicating states collapse into a single absorbing macro state. Where m is the number of states in the macro state, the system of m equations is sent to element (4) for computation of the m values $V(i)$ that can now be computed independently of those for all states not in the macro state. Note that the canonical set that is ergodic in the reduced problem may not have been ergodic in the original problem. Also note that if the problem has no exploitable structure, the entire problem will be sent to element (4) after the first search.

Each search is a search for a null row in the current B matrix, where a null row is one for which $b(i,k)=0$ for every k. Once a set

of states has been evaluated by element (4), the states are added to the set of evaluated states, E , new r_i' values are calculated for the remaining states (those in the set S-E where S is the set of all states), the new current B matrix is defined by deletion of rows and columns, and a new search is made. If E=S , searches terminate and element (4) finally outputs all values V(i) .

The following facts about stochastic matrices provide the basis for the search procedure:

1. State i is absorbing if $b_{ij}=0$ for all j .
2. If state j is absorbing and $b_{ij}=1$, then state i is transient.
3. If state j is transient and $b_{kj}=1$, then state k is also transient.
4. If state i communicates with state j , and j communicates with state k , then i communicates with k .

The details of the algorithm are illustrated in Figure 9. An example problem that illustrates the SVDO procedure follows.

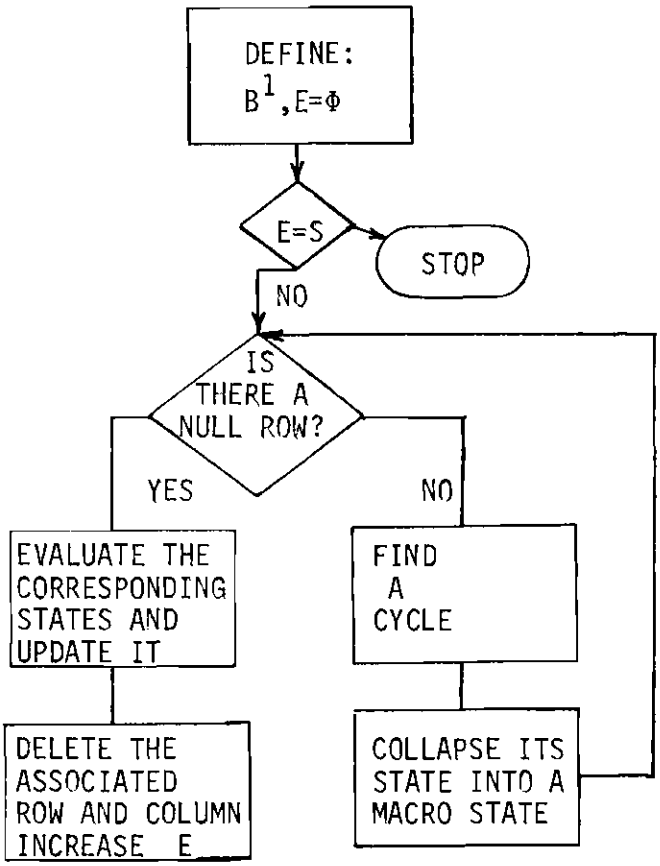
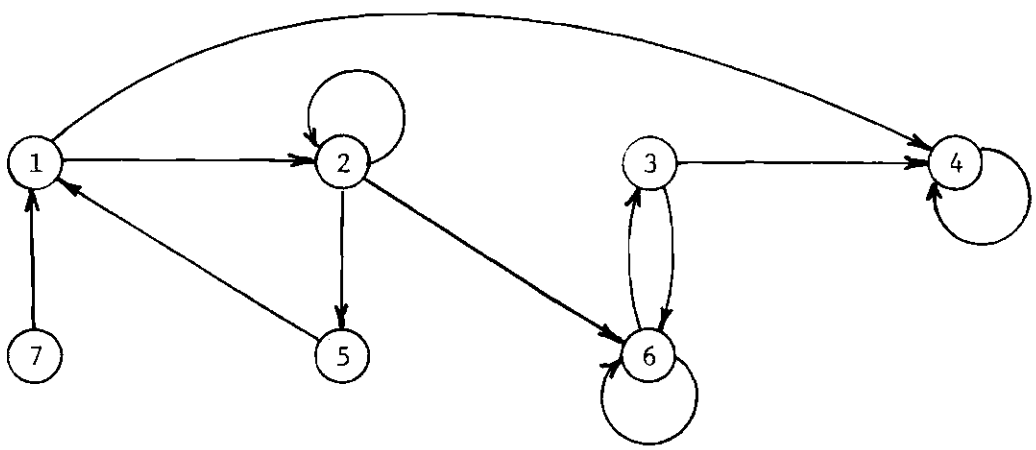


Figure 9. A Flow Chart for SVDO

An Example of SVDO. Consider performing a VDO for a valued Markov chain with the following transition matrix P and associated rewards r^1, where $S=\{1,2,\ldots,7\}$ is the set of all states. At the start the set E of evaluated states is empty. Let $\beta=0.9$.

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & .75 & 0 & .25 & 0 & 0 & 0 \\ 0 & .3 & 0 & 0 & .4 & .3 & 0 \\ 0 & 0 & 0 & .10 & 0 & .9 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .5 & 0 & 0 & .5 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad r^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \\ \begin{bmatrix} 20 \\ 50 \\ 25 \\ 100 \\ 2 \\ 10 \\ 150 \end{bmatrix} \end{array}$$

A transition diagram for the problem is as follows.



SVDO first defines B^1 :

$$B^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Since $E \neq S$, a search is made for a null row, and row 4 is found to be null, indicating that state 4 is absorbing. Thus state 4 can be evaluated immediately, since only $V(4)$ appears on the right-hand side of its VDO equation:

$$V(4) = r^1(4) + \beta p(4,4)V(4) \implies V(4) = 1000 \quad .$$

Since states 1 and 3 lead to 4, as identified by 1's in column 4, $r(1)$ and $r(3)$ are updated:

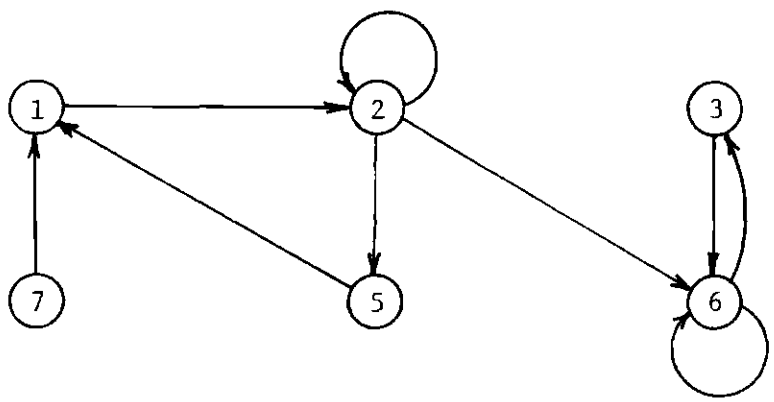
$$r^2(1) = r^1(1) + \beta p(1,4)V(4) = 245 \quad .$$

$$r^2(3) = r^1(3) + \beta p(3,4)V(4) = 115 \quad .$$

Row 4 and column 4 of B^1 are deleted to yield B^2 . The new B^2 and r^2 are:

$$B^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad r^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 7 \end{array} \begin{bmatrix} 245 \\ 50 \\ 115 \\ 2 \\ 10 \\ 150 \end{bmatrix}$$

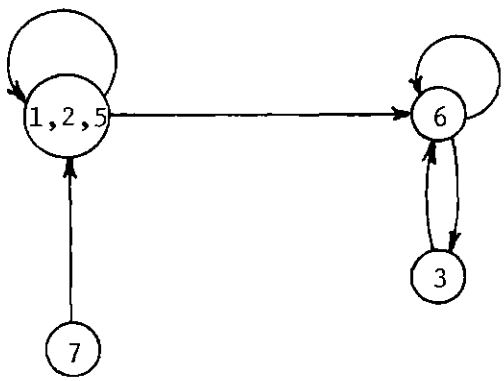
Now $E=\{4\}$. A transition diagram equivalent to B^2 is as follows:



Since $E \neq S$, SMDP looks for a null row in B^2 . None is found. Therefore SMDP looks for a cycle starting with the lowest remaining numbered state, finding the cycle $1 \rightarrow 2 \rightarrow 5 \rightarrow 1$. The rows and columns of the states in the cycle are replaced by their unions, except that $b_{ii}=0$ for any state, including a macro state. We label the new macro state 125. Thus a new incidence matrix B^3 is formed:

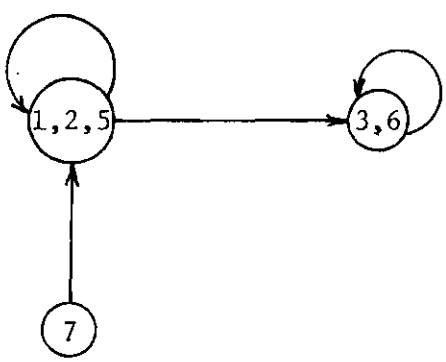
$$B^3 = \begin{array}{c} \\ 125 \\ 3 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccc} \overset{1}{\underset{5}{2}} & 3 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

A transition diagram equivalent to B^3 is as follows:



Since $E \neq S$, a new search is made. No null row is found. The cycle $3 \rightarrow 6 \rightarrow 3$ is found. A new incidence matrix B^4 is formed. This matrix and its equivalent transition diagram are as follows:

$$B^4 = \begin{array}{c} \\ 125 \\ 36 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccc} \overset{1}{\underset{5}{2}} & \overset{3}{\underset{6}{}} & 7 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$



Since $E \neq S$, a new search is made. Row 36 is found to be null, so that states 3 and 6 can be evaluated:

$$V(3) = r^2(3) + \beta p(3,3)V(3) + \beta p(3,6)V(6) = 115 + .81V(6)$$

and $$V(6) = r^2(6) + \beta p(6,3)V(3) + \beta p(6,6)V(6) = 10 + .45V(3) + .45V(6)$$

The solution to this set of equations is $V(3)=384.64$ and $V(6)=332.88$. From the 1 in column 36, it is seen that macro state 125 leads to the state to be deleted, and hence the $r(i)$'s for states 1, 2 and 5 are to be updated.

$$r^3(1) = r^2(1) + \beta p(1,3)V(3) + \beta P(1,6)V(6) = 245$$

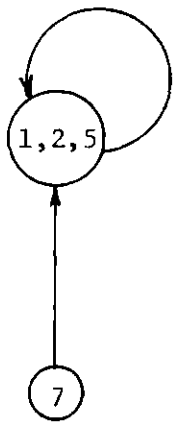
$$r^3(2) = r^2(2) + \beta p(2,3)V(3) + \beta p(2,6)V(6) = 139.88$$

$$r^3(5) = r^2(5) + \beta p(5,3)V(3) + \beta p(5,6)V(6) = 2$$

Now $E=\{4,3,6\}$. A new incidence matrix B^5 is formed:

$$B^5 = \begin{array}{c} \\ 125 \\ 7 \end{array} \begin{array}{c} \begin{matrix} 1 \\ 2 \\ 5 \end{matrix} \quad 7 \\ \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \end{array}$$

A transition diagram equivalent to B^5 is as follows:



The current adjusted rewards for the states that have not been evaluated are as follows:

$$r^3 = \begin{matrix} 1 \\ 2 \\ 5 \\ 7 \end{matrix} \begin{bmatrix} 245 \\ 139.88 \\ 2 \\ 150 \end{bmatrix}$$

Since $E \neq S$, SVDO searches again. A null row is found for the macro state 125, allowing evaluation of states 1, 2 and 5, yielding

$$V(1) = 535.41$$

$$V(2) = 420.23$$

$$V(5) = 483.87$$

State 7 is found to lead to the evaluated macro state, so $r(7)$ is updated to $r^4(7)=631.87$. The row and column corresponding to the macro state 125 is deleted, yielding B^7:

$$B^7 = {}_7\begin{bmatrix} & 7 \\ 0 & \end{bmatrix}$$

The transition diagram equivalent to B^7 is simply one node, node 7. The current adjusted rewards for the states that have not been evaluated consist of one number, $r^4(7)=631.87$. B^7 has a null row, and state 7 is evaluated: $V(7)=631.87$. Now $E=S$, and the VDO is terminated.

The net result of SVDO in this example has been to avoid a "large" 7×7 VDO, replacing it with a series of smaller VDO's: a 1×1 VDO for state 4, a 2×2 VDO for states 3 and 6, a 3×3 VDO for states 1, 2 and 5, and a 1×1 VDO for state 7.

SMDP: A Natural Decomposition Algorithm

SMDP is a decomposition scheme for storable MDP problems with a well defined structure that can be exploited to obtain a sequence of subproblems at each VDO. The details of the decomposition are given in the description of the SVDO algorithm.

In SMDP, once the values are known for a given policy, any of the methods given in Chapter IV for policy iteration can be used. For most problems there is little hope of gaining computational advantage by classifying the states for policy iteration alone, since the problem structure normally changes with each change in the decision policy.

Additionally, the PIR's do not present major computational or storage difficulties.

The SMDP algorithm is described by Figure 10. The framework of Figure 10 is that of Howard's method, although any method that uses VDO's could make use of the SVDO decomposition scheme within SMDP.

DMDP: An Arbitrary Decomposition Algorithm

The Arbitrary-Decomposition Algorithm, DMDP, is a method of decomposing a large problem which is either unstorable and/or has no polydesmic structure, into a sequence of smaller subproblems which are easier to solve. By varying the number of subproblems, a user has the flexibility to vary the trade-off between storage space and computation time, a feature also exhibited by other decomposition algorithms such as that of Dantzig-Wolfe. In addition to the obvious storage advantages, it reduces the problem of inverting a large matrix to inverting several smaller matrices, which can be done not only more rapidly, but with greater numerical stability.

Without loss of generality, the description of the DMDP algorithm and its proof of convergence will be given with reference to Howard's method.

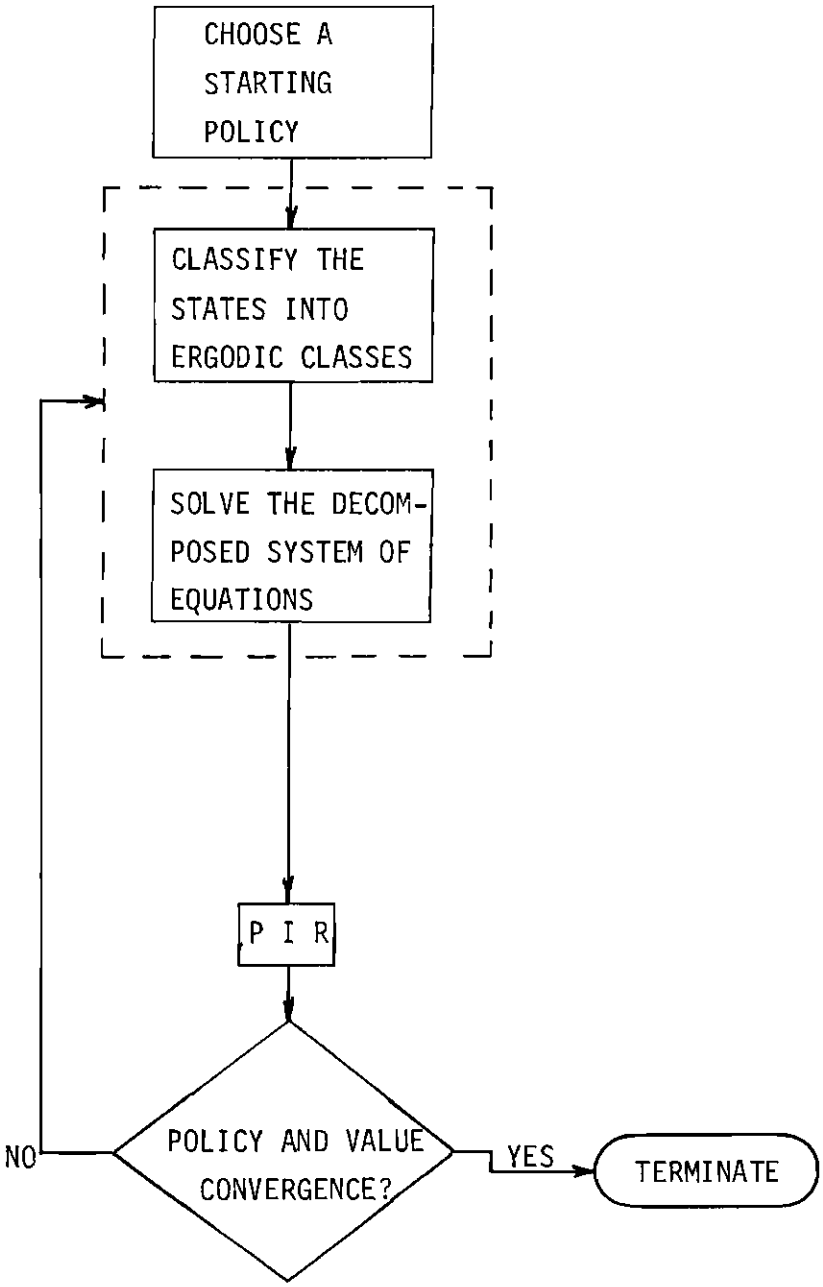


Figure 10. A Flow Chart for SMDP

Description of the Arbitrary-Decomposition Algorithm

In the arbitrary decomposition algorithm, a subset of the variables is given temporarily fixed values and the remaining reduced problem is solved. Then, with a different subset of variables fixed, a new remaining reduced problem is solved. This procedure continues until no improvement is achieved, at which time optimality is guaranteed. The state space may be partitioned into any number of subsets; fine partitioning decreases the size of each subproblem but increases the number of iterations, as compared to coarse partitioning. To simplify the notation, only the details for the partition of the state space into two subsets will be given here. Finer partitions are treated in a straightforwardly analogous manner.

The solution procedure is as follows:

1. In a Markov decision process with states $i=1,2,\ldots,N$, let the state space be partitioned into two subsets, one of size M and the other of size $N-M$. Let the subset $\{V(1),V(2),\ldots,V(M)\}$ be given fixed values, and let Howard's algorithm be used to find the optimal values of the free variables, $\{V(M+1),V(M+2),\ldots,V(N)\}$ under the constraint of the fixed subset of v's:

$$V(i) - \beta\sum_{j=M+1}^{N} p(i,j,k)V(j) = r(i,k) + \beta\sum_{j=1}^{M} p(i,j,k)V(j) \qquad (1)$$

$$i=M+1,M+2,\ldots,N$$

$$k=1,2,\ldots,K(i)$$

Equation (1) is simply the standard value equation with the fixed terms shifted to the right-hand-side, defining a reduced system of linear equations in the non-fixed V 's . Howard's algorithm is applied to this reduced system, with each VDO involving only N-M equations.

2. Now let the subset {V(M+1),V(M+2),...,V(N)} , whose values were the output of step 1, be taken as fixed, and let Howard's algorithm be used to find the optimal values of {V(1),V(2),...,V(M)} under the constraint of the fixed subset:

$$V(i) - \beta\sum_{j=1}^{M} p(i,j,k)V(j) = r(i,k) + \beta\sum_{j=M+1}^{N} p(i,j,k)V(j) \qquad (2)$$

$$i=1,2,\ldots,M$$

$$k=1,2,\ldots,K(i)$$

Here each VDO of Howard's algorithm involves only M equations.

3. The solution procedure alternates between steps 1 and 2 until the same decision policy is encountered in two successive iterations (policy convergence). At this point, if the differences in the v's are asymptotically small (value convergence), optimality has been achieved. Either policy convergence or value convergence can occur first, but optimality is not guaranteed until both are achieved.
4. If policy convergence is achieved first, it is possible,

but wasteful, to continue the same procedure until value convergence is achieved. Hence a termination procedure is used which consists of continuing to alternate between steps 1 and 2 until value convergence is achieved, but omitting all PIR's.

5. Finally, one complete PIR is performed to verify optimality. This can take the form of restoring the disabled PIR routine for one last iteration.

In summary, the solution procedure consists of applying Howard's algorithm alternately to two subproblems until both policy convergence and value convergence are achieved, with intermediate PIR's disabled.

Storage and Efficiency of DMDP. The storage requirement for a VDO is approximately N^2+N for a problem with N states solved directly. When DMDP is used to partition the problem into p subproblems, the storage used in each step is approximately $(N/p)^2+N/p$; thus for large N the storage requirement is reduced by a factor of approximately p^2 . For example, if we partition a large problem into $p=2$ roughly equal parts, the storage requirement is cut by a factor of nearly 4.

There are three effects on computation efficiency. First, data is overwritten p^2 times for each full iteration if DMDP is used, compared to once for all iterations if DMDP is not used. Second, the number of iterations is increased. Third, the approximate number of arithmetic operations for each full iteration is reduced from $(N^3/3)+N^2$ without DMDP to $(N^3/3p^2)+N^2/p$ with DMDP. Computation experience

(see Chapter VII) suggests that, on balance, DMDP should be used only when necessary to save storage space; the total number of arithmetic operations tends to increase by a small amount, and the data overwriting is costly.

Initialization Step

As in all iterative procedures, a good starting solution is of major importance. For the method just described, we have to give initial values for $V(1),V(2),\ldots,V(M)$. The following initialization options may be used:

1. Let $V(1)=V(2)=\cdots=V(M)=0$
2. If we have some idea of the values the $V(i)$'s can take for a given problem, say $v_1,v_2,\ldots,v_M$, let $V(1)=v_1,V(2)=v_2,\ldots,V(M)=v_M$
3. Since $\lim_{\beta\to 1} V(i)=\infty$, and

   $$\lim_{\beta\to 1}\left(V(i)-V(N)\right)=W(i) \qquad i=1,2,\ldots,N$$

   where $W(i)$ is a finite value for all i , the values of the $V(i)$'s, for large values of β , will be of about the same magnitude. It is possible to provide, in lieu of a set of initial values an initial condition, namely, that all values be equal, i.e., let $V(1)=V(2)=\cdots=V(N)$. In this case, the equations for step 1 become

$$V(i) - \beta\sum_{j=M+1}^{N} p(i,j,k)V(j) = r(i,k) + \beta\sum_{j=1}^{M} p(i,j,k)V(N)$$

$$= r(i,k) + \beta V(N)\sum_{j=1}^{M} p(i,j,k)$$

or

$$V(i)-\beta\sum_{j=M+1}^{N-1} p(i,j,k)V(j)-\beta p(i,N,k)V(N)-\beta V(N)\sum_{j=1}^{M} p(i,j,k) = r(i,k)$$

$$V(i)-\beta\sum_{j=M+1}^{N-1} p(i,j,k)V(j)-\beta\Big(p(i,N,k)+\sum_{j=1}^{M} p(i,j,k)\Big)V(N) = r(i,k)$$

$$i=M+1,M+2,\ldots,N$$

For all future iterations, we will have from step 2 values of $V(1),V(2),\ldots,V(M)$ to be used with the original set of equations for step 1.

4. A myopic set of starting values may be obtained from the following facts:

i) $V(i) \simeq g + \beta g + \beta^2 g + \cdots$

$= g(1+\beta+\beta^2+\cdots)$

$= g/1-\beta$

ii) $g \simeq 1/N\big(r(1,k)+r(2,k)+\cdots+r(N,k)\big)$

and in particular,

$$g \simeq 1/N\big(r^*(1)+r^*(2)+\cdots+r^*(N)\big) \quad ,$$

where $r^*(i)=\underset{k}{\text{Max}}\ r(i,k) \quad , \quad i=1,2,\ldots,N$

Thus, we can obtain a good starting solution which approximates the optimal solution as

$$V(1) = V(2) = \cdots = V(M) = \frac{1/N\sum_{i=1}^{N} r^*(i)}{1-\beta}$$

Some other solutions may be used for a particular problem. Option 1 and 2 are the easiest to implement but Option 2 is not always available. Option 3 involves some re-programming to treat the first iteration separately. Option 4, which seems to be superior for most problems, requires only a few preliminary computations.

An Example

To illustrate the DMDP method, we will solve Howard's "Taxicab Problem" [55], which he states as follows: Consider the problem of a taxicab driver whose territory encompasses three towns, A , B , and C . If he is in town A , he has three alternatives:

1. He can cruise in the hope of picking up a passenger by being hailed.
2. He can drive to the nearest cab stand and wait in line.
3. He can pull over and wait for a radio call.

If he is in town C , he has the same three alternatives, but if he is in town B , the last alternative is not present because there is no radio cab service in that town. For a given town and given alternative, there is a probability that the next trip will go to each of the towns A , B , and C and a corresponding reward in monetary units associated with each such trip. This reward represents the income from the trip after all necessary expenses have been deducted. For example, in the case of alternatives 1 and 2, the cost of cruising and of driving to the nearest stand must be included in calculating the rewards. The probabilities of transition and the rewards depend upon the alternative because different customer population will be encountered under each alternative.

If we identify being in towns A , B , and C with states 1, 2, and 3, respectively, then we have the following information:

State	Alternative	Probability			Reward			Expected Immediate Reward
i	k	p(i,j,k)			c(i,j,k)			r(i,k)
		j=1	2	3	j=1	2	3	
1	1	1/2	1/4	1/4	10	4	8	8
	2	1/16	3/4	3/16	8	2	4	2.75
	3	1/4	1/8	5/8	4	6	4	4.25
2	1	1/2	0	1/2	14	0	18	16
	2	1/16	7/8	1/16	8	16	8	15
3	1	1/4	1/4	1/2	10	2	8	7
	2	1/8	3/4	1/8	6	4	2	4
	3	3/4	1/16	3/16	4	0	8	4.5

The state space for this example was partitioned into two subsets with states 1 and 2 in the first subset. The value equations under this partition become:

Subproblem #1

$$V(i) - \beta \sum_{j=1}^{2} p(i,j,k)V(j) = r(i,k) + \beta p(i,3,k)V(3) \quad , i=1,2$$

and

Subproblem #2

$$V(3) - \beta p(3,3,k)V(3) = r(3,k) + \beta \sum_{j=1}^{2} p(3,j,k)V(j)$$

or

$$V(3) = \frac{r(3,k) + \beta \sum_{j=1}^{2} p(3,j,k)V(j)}{1 - \beta p(3,3,k)}$$

Howard's method was used in subproblem #1 to find the best V(1) and V(2) to go with the given value of V(3), and in subproblem #2, the best V(3) corresponding to the fixed values of V(1) and V(2) was found using the above equation in a single policy iteration.

The problem was solved with two different initial values of V(3), and the solution for the entire problem was computed using Howard's method. The results are given in Tables 10, 11, and 12.

For the results given in Table 11, the initial value of V(3) was 0, as in option 1 of the initialization step, and a δ-convergence of 0.10 was obtained in 7 full iterations of the DMDP algorithm, where by an iteration we mean the solution of the two subproblems. Nine VDO's for a system of size 2×2 (60 arithmetic operations) were performed using the decomposition method, while 3 VDO's for a system of size 3×3 (54 arithmetic operations) were required when solving the problem as a whole. Note that the values of all the V(i)'s increased from one iteration to the next.

For those results of Table 12, the initial value of V(3) was 200, as in option 2 of the initialization step, and a δ-convergence of 0.10 was obtained in 8 iterations of the DMDP algorithm. Eleven VDO's for a system of size 2×2 were performed using the decomposition method. Here, the values of all the V(i)'s decreased from one iteration to the next.

Table 10. Howard's "Taxicab Problem"
Solved by Howard's Method

n	V(1)	k	V(2)	k	V(3)	k	
0		1		1		1	INITIALIZATION
1	91.26	1	97.55	1	89.97	1	VDO
2	91.26	1	102.02	2	90.24	2	PIR
3	119.44	1	134.48	2	121.93	2	VDO
4	120.82	2	134.48	2	121.93	2	PIR
5	121.65	2	135.31	2	122.84	2	VDO
6	121.65	2	135.31	2	122.84	2	PIR

Table 11. Howard's "Taxicab Problem"
Solved by DMDP (V(3)=0)

	n	V(1)	k	V(2)	k	
V(3)=0 R(1,1)= 8 R(1,2)= 2.75 R(1,3)= 4.25 R(2,1)=16 R(2,2)=15	0					INITIALIZATION
	1	25.849581	1	27.632311	1	VDO
	2	25.849581	1	38.214483	2	PIR
	3	48.69565	1	83.478257	2	VDO
	4	61.836953	2	83.478256	2	PIR
	5	65.872173	2	88.024983	2	VDO
	6	65.872171	2	88.024982	2	PIR
V(3)=79.805612,k=2 R(1,1)=25.956262 R(1,2)=16.217197 R(1,3)=49.140655 R(2,1)=51.912525 R(2,2)=19.489065	7					PIR
	8	102.11258	2	118.74305	2	VDO
	9	102.11258	2	118.74304	2	PIR
V(3)=107.7625,k=2 R(1,1)=32.246562 R(1,2)=20.934922 R(1,3)=64.866405 R(2,1)=64.493124 R(2,2)=21.061641	10					PIR
	11	114.80805	2	129.50396	2	VDO
	12	114.80805	2	129.50396	2	PIR
V(3)=117.55614,k=2 R(1,1)=34.450131 R(1,2)=22.587598 R(1,3)=70.375327 R(2,1)=68.900263 R(2,2)=21.612533	13					PIR

Table 11. Continuation

	n	V(1)	k	V(2)	k	
	14	119.25542	2	133.27364	2	VDO
	15	119.25543	2	133.27364	2	PIR
V(3)=120.98698,k=2 R(1,1)=35.22207 R(1,2)=23.166553 R(1,3)=72.305175 R(2,1)=70.44414 R(2,2)=21.805517	16					PIR
	17	120.8134	2	134.59421	2	VDO
	18	120.8134	2	134.59421	2	PIR
V(3)=122.18884,k=2 R(1,1)=35.492489 R(1,2)=23.369366 R(1,3)=72.981221 R(2,1)=70.984978 R(2,2)=21.873122	19					PIR
	20	121.35917	2	135.05682	2	VDO
	21	121.35917	2	135.05682	2	PIR
V(3)=122.60987,k=2 R(1,1)=35.587221 R(1,2)=23.440415 R(1,3)=73.21805 R(2,1)=71.174441 R(2,2)=21.896805	22					PIR
	23	121.55037	2	135.21888	2	VDO
	24	121.55037	2	135.21888	2	PIR
V(3)=122.75736,k=2 R(1,1)=35.620406 R(1,2)=23.465304 R(1,3)=73.301014 R(2,1)=71.240811 R(2,2)=21.905101	25					PIR

Table 12. Howard's "Taxicab Problem"
Solved by DMDP (V(3)=200)

	n	V(1)	k	V(2)	k	
V(3)=200 R(1,1)= 53.999999 R(1,2)= 36.5 R(1,3)=116.75 R(2,1)=106.99999 R(2,2)= 26.25	0					INITIALIZATION
	1	171.25347	1	183.06406	1	VDO
	2	175.87673	3	183.06406	1	PIR
	3	177.63588	3	185.93615	1	VDO
	4	177.63589	3	185.93615	1	PIR
V(3)=168.44049,k=2 R(1,1)= 45.89911 R(1,2)= 31.174333 R(1,3)= 98.997774 R(2,1)= 91.798221 R(2,2)= 24.474777	5					PIR
	6	150.92331	3	159.71371	1	VDO
	7	150.92331	3	159.71371	1	PIR
V(3)=145.11056,k=2 R(1,1)= 40.649876 R(1,2)= 27.237407 R(1,3)= 85.874688 R(2,1)= 81.299751 R(2,2)= 23.162469	8					PIR
	9	131.1764	3	140.32913	1	VDO
	10	131.25331	1	141.05033	2	PIR
	11	132.89023	1	144.17667	2	VDO
	12	132.89023	1	144.17667	2	PIR
V(3)=131.00778,k=2 R(1,1)= 37.47675 R(1,2)= 24.857563 R(1,3)= 77.941875 R(2,1)= 74.953501 R(2,2)= 22.369188	13					PIR

Table 12. Continuation

	n	V(1)	k	V(2)	k	
	14	124.70765	1	138.27761	2	VDO
	15	125.20975	2	138.27761	2	PIR
	16	125.36393	2	138.45133	2	VDO
	17	125.36393	2	138.45133	2	PIR
V(3)=125.69925,k=2 R(1,1)= 36.282331 R(1,2)= 23.961748 R(1,3)= 74.955827 R(2,1)= 72.564662 R(2,2)= 22.070583	18					PIR
	19	122.95328	2	136.40802	2	VDO
	20	122.95328	2	136.40802	2	PIR
V(3)=123.83961 R(1,1)= 35.863912 R(1,2)= 23.647934 R(1,3)= 73.90978 R(2,1)= 71.727824 R(2,2)= 21.965978	21					PIR
	22	122.1088	2	135.69222	2	VDO
	23	122.1088	2	135.69222	2	PIR
V(3)=123.18815,k=2 R(1,1)= 35.717334 R(1,2)= 23.538 R(1,3)= 73.543333 R(2,1)= 71.434667 R(2,2)= 21.929333	24					PIR
	25	121.81299	2	135.44147	2	VDO
	26	121.81299	2	135.44146	2	PIR
V(3)=122.95994,k=2 R(1,1)= 35.665986 R(1,2)= 23.49949 R(1,3)= 73.414965 R(2,1)= 71.331972 R(2,2)= 21.916496	27					PIR
	28	121.70934	2	135.35363	2	VDO
	29	121.70934	2	135.35363	2	PIR

Table 12. Continuation

	n	V(1)	k	V(2)	k	
V(3)=122.88 , k=2	30					PIR
R(1,1)= 35.648						
R(1,2)= 23.486						
R(1,3)= 73.369999						
R(2,1)= 71.296						
R(2,2)= 21.912						

Convergence of DMDP. Let us define a class of simplex-related solution procedures for any primal linear program whose variables are grouped into groups that we will call j-groups. In the MDP linear programs, each j-group is the group of variables corresponding to a given state j, and there are as many variables in a j-group as there are decisions $k \varepsilon K_j$ for the state (see the linear programming formulation in Chapter IV). The class of simplex-related solution procedures is the class in which there are Z different j-groups that are free to have their basic variable changed in one block-pivot or multiple-basis-entry iteration. If $Z=N$, we have the h-algorithm described below, which is Howard's algorithm in primal space. If $Z=1$, we have the ℓ-simplex algorithm described below. These two algorithms are special cases of DMDP. For maximal decomposition into N subproblems, one for each state, DMDP is the ℓ-simplex algorithm in dual space; for minimal decomposition into one subproblem, DMDP is (identically) Howard's algorithm. For all Z, $1 \leq Z \leq N$, corresponding to all versions of DMDP, the class of simplex-related procedures does converge. This follows immediately from the fact that convergence proofs for the simplex algorithm do not depend on the simplex algorithm's requirement that the variable added to the basis be that with the largest absolute cost coefficient among those of indicated sign, but only that some variable among those indicated be added at each iteration.

In DMDP, if the problem is decomposed into p subproblems, at most p iterations will be performed before a given state is free to have its primal variable changed. Thus, at worst, p iterations of DMDP may be required to make the same decision changes (changes of

basic primal variable) as are made in one iteration of Howard's algorithm.

The ℓ-simplex algorithm is defined as a linear programming procedure identical to the simplex algorithm except that the leftmost variable whose cost coefficient is of the indicated sign is added to the basis. In dual space, the equivalent algorithm is a procedure identical to the dual-simplex algorithm except that the topmost constraint whose right-hand side is negative dictates the pivot. This procedure is identical to DMDP with N subproblems, provided that the free states are rotated in the order $1,2,\ldots,N,1,2,\ldots$.

For DMDP in general, we see that our description is identical step-by-step to the description of Howard's algorithm in Chapter IV, except that in Step 3 we do not find the worst-violated constraint in each group, but simply find a violated constraint among the free group if one exists, and continue to rotate the free groups until any violated constraints will be found in at most p iterations.

Let the value convergence rate of a given algorithm, $VCR(\cdot)$, be defined by

$$VCR(\cdot) = \left| \frac{V^n(i) - V^{n-1}(i)}{V^{\infty}(i) - V^{n-1}(i)} \right|$$

for some test state i. (The notation is that of Chapter IV.) Experimental evidence shows that Howard's algorithm has a faster convergence rate than the simplex algorithm or the dual simplex algorithm when applied to MDP problems. Thus we can expect that

$$VCR(\ell\text{-simplex}) \leq VCR(DMDP) \leq VCR(\text{Howard's algorithm}) \quad .$$

This expectation is verified by the computational experience given in Chapter VII, and in fact the convergence rate increases uniformly with decreasing p.

A General Macro Solution Framework

For a given computer, there is a largest number of states, N^*, above which direct solution of an MDP problem becomes substantially more costly. This is commonly due to storage limitations, but it may also stem from software limitations such as dimension limits on routines for simultaneous solution of linear systems of equations. If there is no natural structural basis for decomposition of a larger problem, it is advantageous to partition the problem arbitrarily, but not into more pieces than are required to get below N^* in the size of any one piece. Because computation time increases more than linearly with subproblem size for practically any solution method, the sizes of subproblems generated arbitrarily should be roughly equal. If there is a natural structure for a problem of size larger than N^*, it is advantageous to partition the problem naturally. Subproblems generated arbitrarily may (for some decision policies at least) have a natural structure allowing for advantageous natural decomposition. These ideas lead to the proposed way of combining SMDP and DMDP that is incorporated in Figure 11.

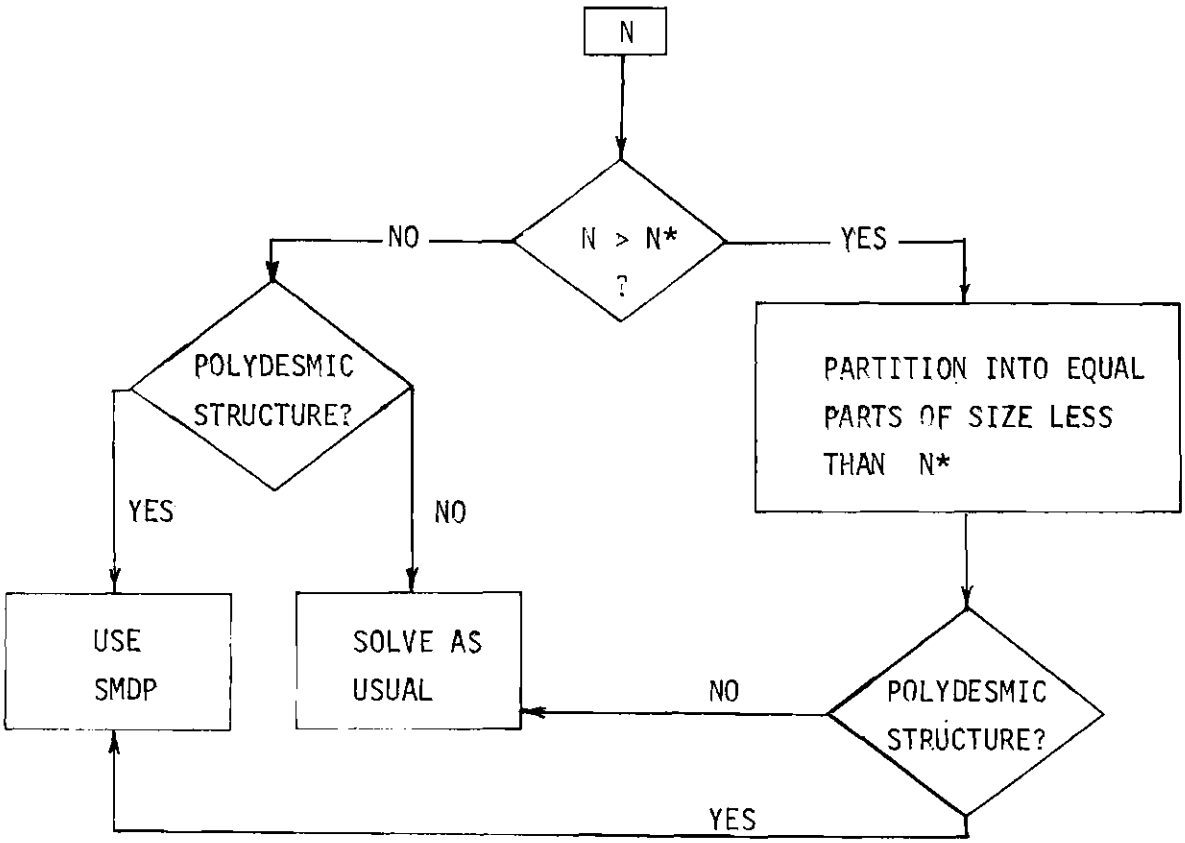


Figure 11. Recommended Way of Selecting Macro Solution Method

CHAPTER VII

COMPUTATIONAL EXPERIENCE WITH PROPOSED METHODS

This chapter presents two experiments that test the applicability of the new solution methods advanced in this thesis. The first experiment analyzes the computational load of the three most promising of the new methods, as compared to that of existing methods, using large-scale problems randomly generated to cover the range of reasonable variations in problem data. The second experiment is similar, except that the problem solved is a large-scale inventory-control problem, so that the data are generated from a realistic problem situation.

Proposed Large-Scale Methods

Small-scale computational experience on various techniques, as reported in previous chapters, suggests that three specific methods are worthy of serious consideration for reducing the computational load in solving large-scale Markov decision problems. To avoid confusion among the many versions of various methods discussed earlier, the specific new methods used in the experiments reported here are redescribed as follows.

The Forecast Acceleration technique is the 2-V forecast-acceleration technique of Equation (10), Chapter IV, starting with $V(i)=\bar{r}/(1-\beta)$ for all i, where $\bar{r}$ is the mean of all $r(i,k)$ under the current policy. Jumps to a forecast set of $V(i)$ values were made whenever,

for all i, two successive forecasts yielded values within 15 percent of each other. The iterations were terminated upon δ-convergence with $\delta=0.1$, so that all $V(i)$ were within about 1.0 of their true values upon termination.

The Arbitrary Decomposition technique is DMDP as described in Chapter VI and listed in Appendix III, with Option 4 of the initialization step (identical to the initialization described above for the Forecast-Acceleration technique). For both experiments, the state space is arbitrarily decomposed into two equal parts.

The Natural Decomposition technique is SMDP as described in Chapter VI and listed in Appendix IV.

Comparison Techniques

In the first experiment, whose purpose is to compare computational loads of the Forecast Acceleration technique, the Arbitrary Decomposition technique and the Natural Decomposition technique against those of standard methods, the mathematical operation of interest is the performance of a single VDO (value determination operation). This is because the new techniques do not differ significantly from their standard counterparts except in their method of performing VDO's, and because the computational load of PIR's, internal data manipulation and other operations are small compared to that of VDO's. Thus the first experiment consists of comparing the three new techniques with the standard in performance of a single VDO--the calculation of $V(1)$, $V(2),\ldots,V(N)$ for a given policy. In the second experiment, whose purpose is to compare the new techniques against standard methods in

solution of a realistic MDP problem, the entire problem is solved by all methods.

The chosen standard method for comparison in both experiments is Howard's method, with standard Gauss-Jordan solution of VDO's. This is because previously-reported improvements in micro solution procedures (see Chapter IV) are equally applicable to the new techniques as to existing techniques, and because the claimed reductions in computational load for the new techniques significantly exceed those claimed for previously-reported techniques.

Random test problems for the first experiment were generated by the procedure described in Appendix V, which is capable of producing Markov transition matrices with any number of equivalence classes, each with a given number of recurrent states or a given number of transient states. The procedure is also capable of randomly sorting the rows and columns or of leaving them in natural order, so that every possible structure of a Markov transition matrix can be generated.

An Experiment to Compare Large-Scale Computational Loads

In the first experiment, two replications of each of six randomly-generated test problems of size 100×100 were used. A VDO, with $\beta=0.9$, was performed by each method on each replication, both with the matrix in random order and with the matrix in natural (canonical) order. Each test problem contains from 1 to 100 "blocks" of states, where a block is an equivalence class with recurrent states or an equivalence class with transient states, as shown in Table 13:

Table 13. Structure of Test Problems

Number of classes with recurrent states	Number of classes with transient states	Number of states in each class	Number of blocks
1	0	100	1
1	1	50	2
5	5	10	10
10	10	5	20
25	25	2	50
50	50	1	100

As can be seen from Table 13, the test problems range from completely ergodic (one block) to maximally structured (100 blocks). In all the test problems except the one-block problem, the number of blocks containing recurrent states and the number of blocks containing transient states are equal. The structure of a typical transition matrix is shown in Figure 12, where each square represents a 10×10 block of mostly non-zero elements and each row of asterisks represents a group of 10 rows with mostly non-zero elements (blank portions represent zero elements).

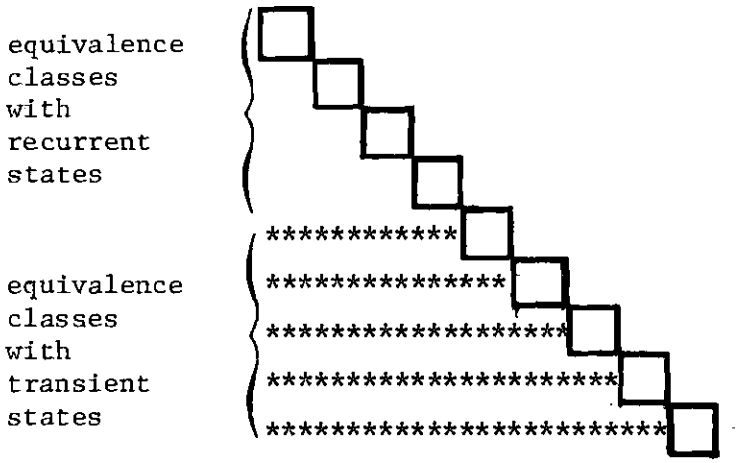


Figure 12. Structure of Transition Matrix for 10-Block Problem

Total execution time in seconds, to the nearest millisecond, was measured on a Univac 1108 computer for both replications of each problem, both with the matrix presented in the natural order shown in Figure 12 and with the matrix presented in random order. The computer programs used to solve the problems are those listed in Appendix I; they vary from one solution method to the other only to the minimal extent necessary. It was judged that the only significant portion of total execution time not used in actual solution of the VDO was that used in reading the elements of the transition matrix P element-by-element from a file with 10,000 rows, which was the same in every replication and every method.

The results of the experiment are given in Table 14 and shown graphically in Figures 13, 14, 15 and 16.

From Figure 13 it is seen that standard Gauss-Jordan solution of the 100-state VDO took approximately 22 seconds of execution time

Table 14. Computation Times for 100-State Test Problems

Number of Blocks:	Standard Gauss-Jordan (GJVDO)		Forecast Acceleration (FAVDO)		Natural Decomposition (SVDO)		Arbitrary Decomposition (DVDO)	
	Random	Non-Random	Random	Non-Random	Random	Non-Random	Random	Non-Random
1	23.194	21.910	20.628	14.198	23.387	23.392	64.090	62.636
	22.874	22.258	19.749	13.584	24.910	23.521	58.555	41.971
2	25.557	21.959	18.196	14.908	15.916	14.994	58.139	21.551
	33.732	22.385	17.892	14.359	15.954	15.560	57.468	21.682
10	35.442	21.725	30.935	19.681	13.061	12.735	71.344	20.955
	35.991	22.130	19.241	17.333	12.993	12.448	63.570	21.438
20	25.932	21.995	30.748	19.212	12.291	12.510	81.580	22.415
	22.537	21.910	25.875	19.256	12.806	12.485	74.290	22.491
50	22.790	22.835	30.840	19.530	13.249	13.187	108.641	20.819
	23.536	22.656	23.676	19.356	13.329	13.048	113.183	22.571
100	22.529	22.135	18.489	12.215	12.054	11.564	20.117	18.256
	22.417	22.609	19.290	12.950	11.940	11.189	19.926	17.196

Note: Computation times are CPU seconds on a Univac 1108 computer for performing a single VDO using the indicated computer program. Each row represents a single randomly-generated 100-state problem which was presented to each of the four programs in two versions--one with the rows and columns randomly sorted ("random") and the other with the rows and columns left in natural order ("non-random").

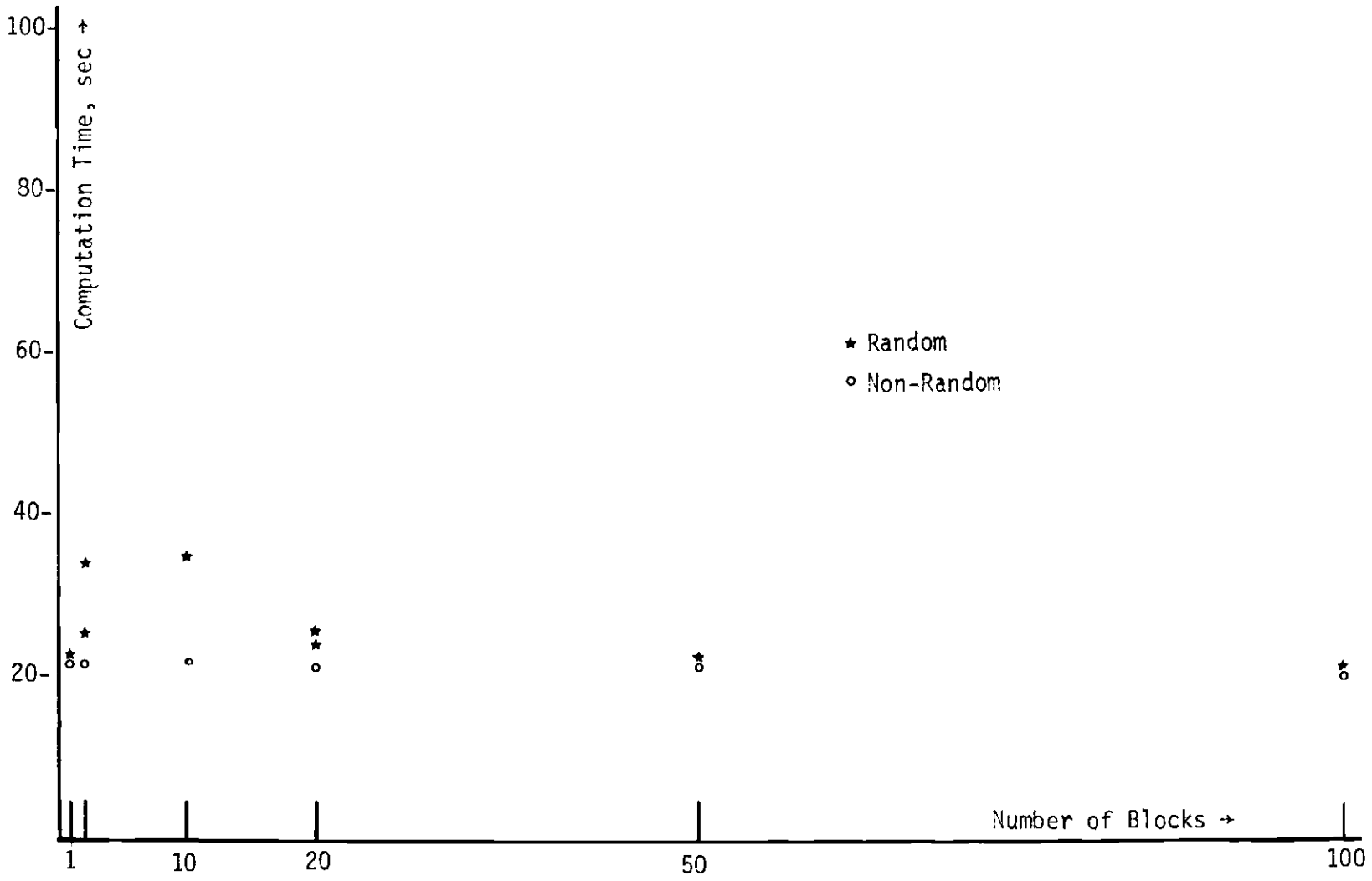


Figure 13. Computation Time for Standard Gauss-Jordan Solution (GJVDO)

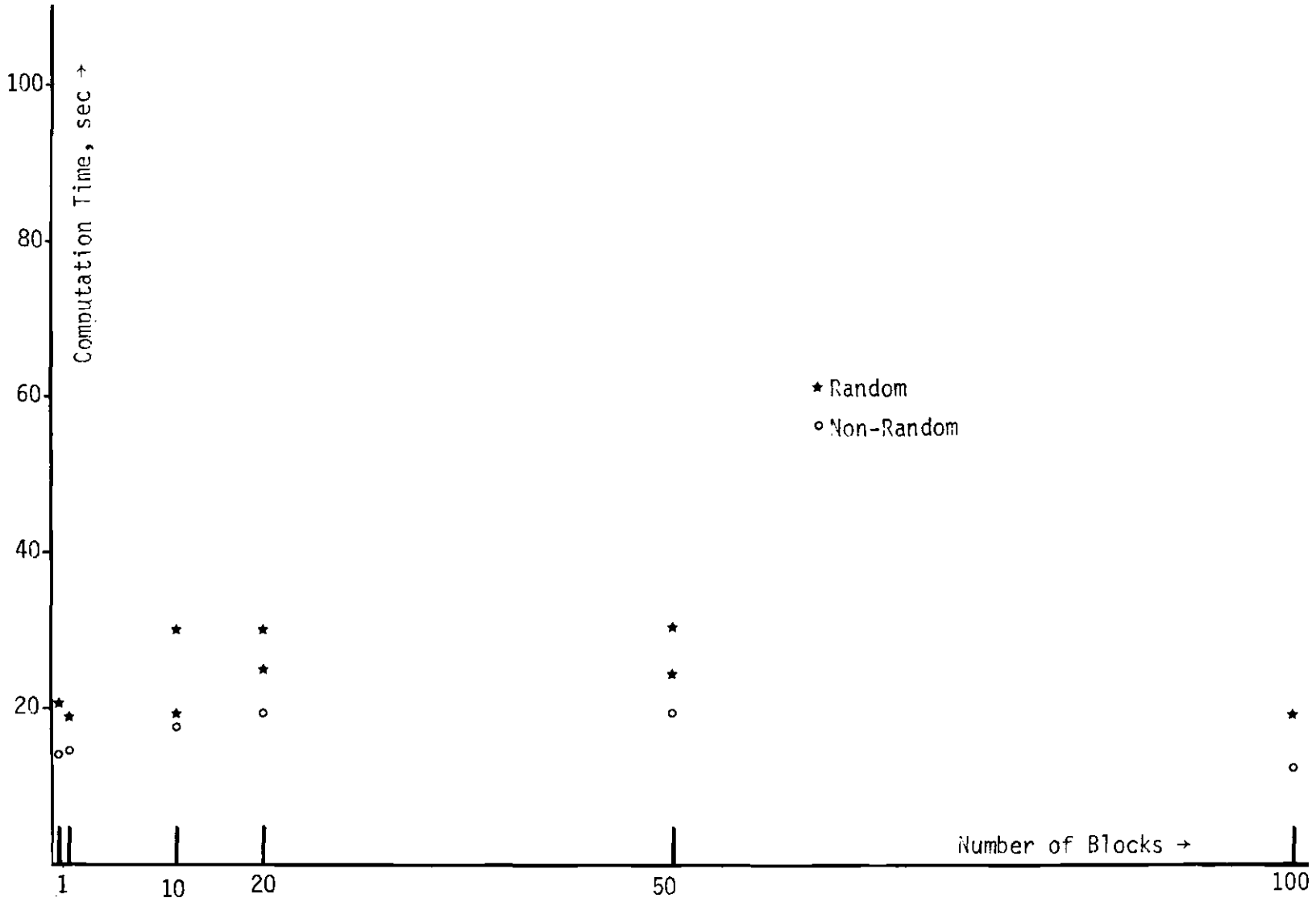


Figure 14. Computation Time for Forecast Acceleration (FAVDO)

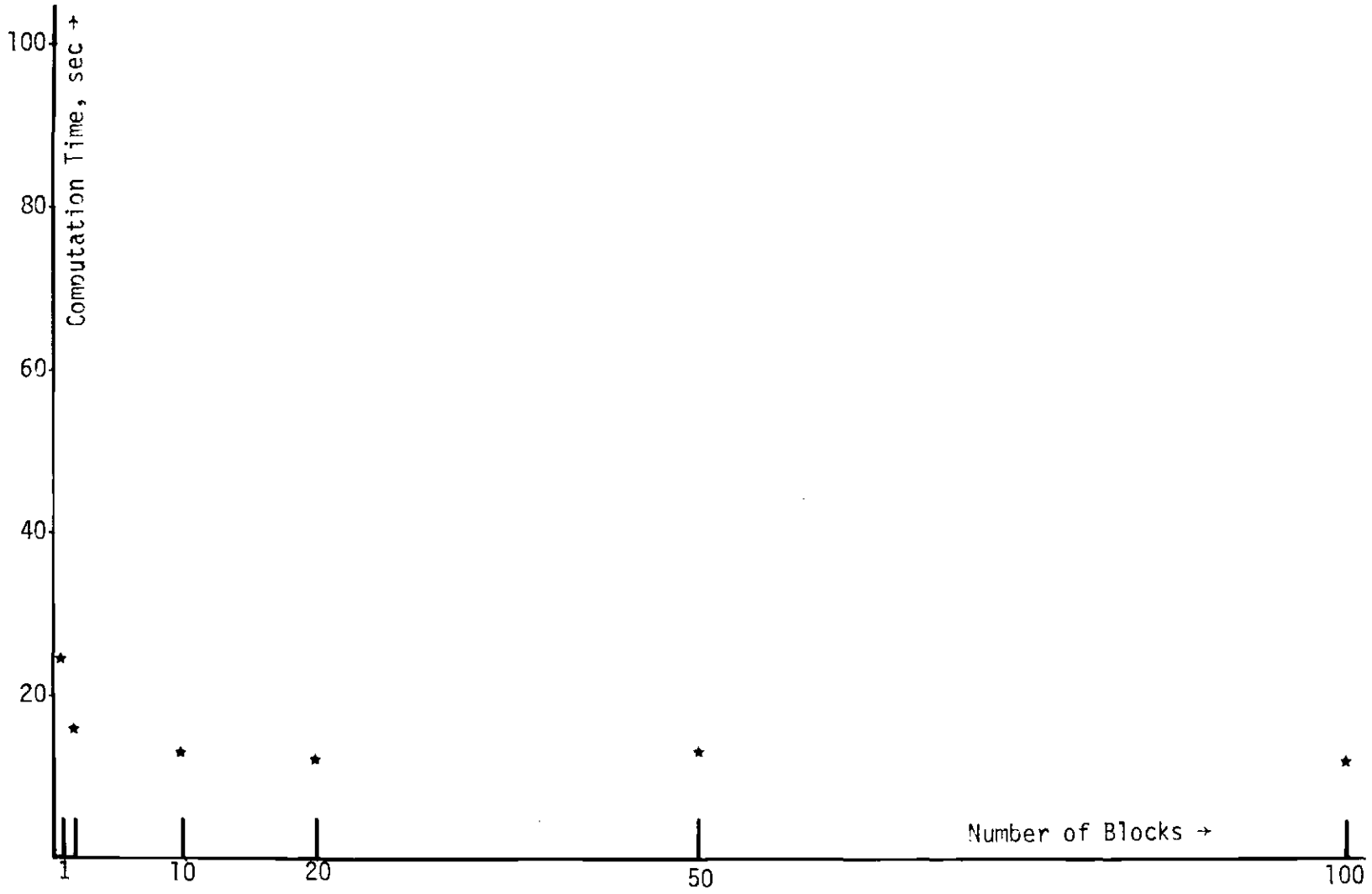


Figure 15. Computation Time for Natural Decomposition (SVDO)

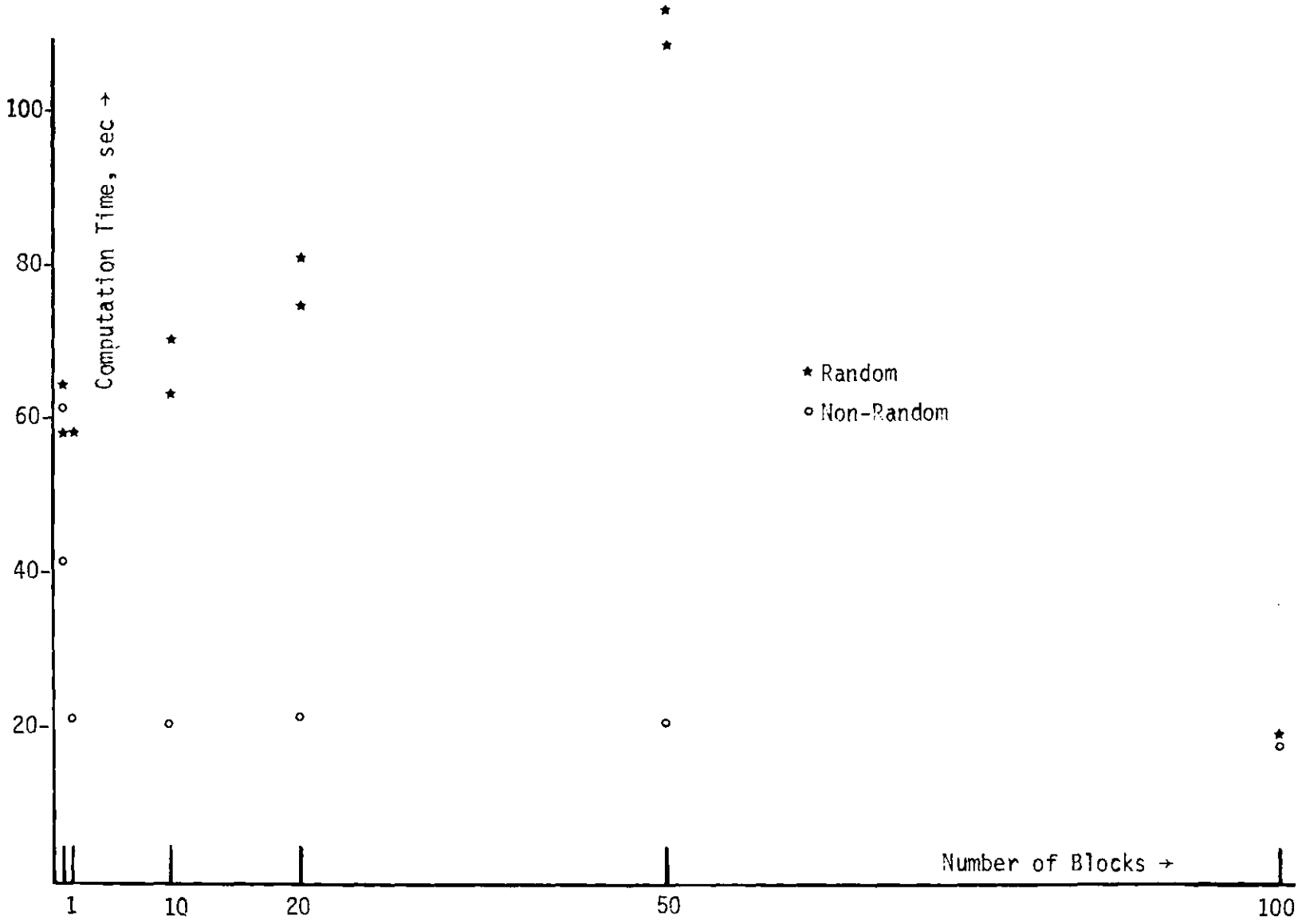


Figure 16. Computation Time for Arbitrary Decomposition (DVDO)

regardless of the structure of the matrix when the matrix was presented in natural order. When the order of the rows and columns was randomized, the solution times increased somewhat because of the lesser diagonal dominance of the matrix when randomized; the increase was negligible for the very sparse matrices of 50 and 100 blocks and for the completely ergodic matrix of one block. The solution times for this method give the standard of comparison against which to judge the three new techniques.

From Figure 14 it is seen that the Forecast Acceleration technique gave solution times comparable to those of the standard Gauss-Jordan method. Numerous spot checks indicated that jumps tended to occur an average of 5 to 15 iterations apart, that each jump had roughly the effect of 10 iterations (β=0.9 in all problems), and that the number of iterations was cut roughly in half as compared to solution of the same problems without forecast acceleration. A 10-block problem was run with the Forecast Acceleration feature disabled in order to simulate classical successive approximation, and the solution was terminated after 40 seconds without convergence; a rough extrapolation indicated convergence would have been achieved in the neighborhood of a total of 70 seconds.

For comparison, we note that Hitchcock and MacQueen [53] found their improved successive approximation to give equal computation times with the Gauss-Jordan method for problem sizes of roughly $N=300$, and their results experimentally affirmed the notion that successive-approximation computation time is approximately proportional to N^2 whereas

Gauss-Jordan computational time is approximately proportional to N^3. A simple ratio calculation indicates that Hitchcock and MacQueen's improved successive approximation would have used an average of about 60 seconds of computation time on the VDO's of the first experiment, which is somewhat less than the 70 seconds indicated above for classical successive approximation.

Given the relative rates of increase in computation time with N, the results shown in Table 14 directly establish Forecast Acceleration as faster than the Gauss-Jordan technique for problems of sizes as low as N=100. Indirect comparison with the results of Hitchcock and MacQueen indicates also that Forecast Acceleration is much faster than their improved successive-approximation procedure.

Figure 15 shows that the Natural Decomposition technique exhibits remarkably low and remarkably constant solution times on the order of 13 seconds for all problems except the one-block (ergodic) problems where there was no natural structure. Since the natural-decomposition code sends the entire problem to the Gauss-Jordan subroutine when no structure is found, the Gauss-Jordan and Natural Decomposition methods are identical for a one-block problem except for the time used by the Natural Decomposition method to find that there is no structure; comparing the one-block computation times from Table 14, we see that this time is negligible.

Figure 16 shows that the Arbitrary Decomposition technique is extremely sensitive to the amount of natural structure when the matrix is presented in random order, but is equivalent to Gauss-Jordan except for large times in the one-block (ergodic) problems when the matrix is

presented in natural order. The phenomena represented in Figure 16 are less complex than they appear at first glance. Recall that in this method the state space is decomposed into two equal parts, so that in this experiment the arbitrary structure coincidentally matches the natural structure. For the 10-block problem diagrammed in Figure 12, for example, both of the decomposition methods first solve the first 50 states as one unit. Thus the low computation times are partly an artifact of the experiment structure.

Another interesting result of the arbitrary-decomposition experiment is the severe upward trend in computation times as the natural structure increases, when the matrix is presented in random order. Recall that the method was designed to handle dense matrices, and that the density decreases as the number of blocks increases. With half of the values held constant, the right-hand side of the $V(i)$ equation often contains so few values $V(j)$ that very little convergence is achieved in a given computation. The 100-block problem, on the other hand, has half of its equations in the form $V(i)=\text{constant}$, so rapid convergence is achieved.

In summary, the first experiment indicates the following regimes of applicability:

> Standard Gauss-Jordan for MDP problems (or subproblems) with less than 100 states
>
> Natural Decomposition for MDP problems with more than 100 states, except for completely ergodic problems
>
> Forecast Acceleration for completely ergodic MDP problems (or subproblems) with more than 100 states
>
> Arbitrary Decomposition only for MDP problems so large that no other methods can be applied

An Experiment on a Large-Scale Inventory Problem

In the second experiment, whose purpose is to compare the application of the three new methods against standard methods in solving a realistic problem, a 100-state inventory control problem was solved. The problem chosen was a large version of the non-linear inventory control problem presented as the last example in Chapter II.

Data for the problem are as follows:

Maximal number of items:	100 items
Demand distribution:	Poisson with mean 50
Shortage cost:	\$100 per item
Manufacturing cost:	\$50 for first item, \$4 per item up to 49 items, \$100 per item above 49 items
Holding cost:	\$0 per item for first 33 items, \$5 per item for next 33 items, \$3 per item above 66 items

With these data the optimal solution is as shown in Table 15. The solution is not of the usual (s,S) type because of the non-linearities in the costs. In particular, the manufacturing costs here include not only a setup cost, but also an "overtime" increase; and the holding costs have three ranges, as might be encountered where there is an existing low-cost storage facility of limited capacity, with storage whose unit cost decreases when a sufficiently large excess is stored.

Solution Methods Tested

For comparison purposes the problem was solved by Howard's algorithm using standard Gauss-Jordan solution of the VDO's. The Arbitrary Decomposition and Natural Decomposition methods, which are basically methods for doing VDO's, were tested by having Howard's algorithm call

Table 15. Inventory Problem Solution

Entering Inventory	Optimal Policy	Expected Discounted Sum of Costs, $
0	Manufacture 49 , increasing stock to 49	3384
1	49 50	3320
2	49 51	3264
⋮	⋮ ⋮	⋮
x	49 49+x	
⋮	⋮ ⋮	⋮
30	49 79	2685
31	49 80	2680
32	49 81	2675
33	48 81	2671
34	47 81	2667
⋮	⋮ ⋮	⋮
x	81-x 81	
⋮	⋮ ⋮	⋮
69	12 81	2527
70	11 81	2523
71	0 71	2515
72	0 72	2503
⋮	⋮ ⋮	⋮
x	0 x	
⋮	⋮ ⋮	⋮
100	0 100	2427

subroutines that solved the VDO's by these methods. The Forecast Acceleration technique, on the other hand, is basically an improvement of White's successive-approximation algorithm, and was tested by including forecast acceleration in White's method, which has no VDO's. All computer codes for the three new methods were the same as that used in the first experiment.

Results

All methods gave the solution shown in Table 15. The execution times are as shown in Table 16.

Table 16. Execution Times for Inventory Problem

Solution Method	No. of VDO's	Execution Time, sec Total	per VDO
Howard's Algorithm with Standard VDO's	5	96.6	19.3
Howard's Algorithm with Arbitrary Decomposition	5	81.9	16.4
Howard's Algorithm with Natural Decomposition	5	52.6	10.46
White's Algorithm with Forecast Acceleration	0 (6 policy improvements)	77.4	-

Using Howard's algorithm with standard Gauss-Jordan VDO's, the problem was solved in 96.6 seconds of total execution time. Five VDO's were performed, so that the execution time per VDO (ignoring the small time devoted to input, output and PIR's) was 19.3 seconds. This compares favorably to the average VDO time of 22 seconds found in the first experiment when the same method was tested on randomly-generated VDO's presented in natural order, showing that the inventory problem's high degree of diagonal dominance makes it basically an easier problem to solve than the randomly-generated problems were.

Using Howard's algorithm with Arbitrary Decomposition VDO's, the problem was solved in 81.9 seconds of total execution time, or 16.4

seconds per VDO. This is approximately a 15 percent reduction in time compared to that of the standard method. Comparison with the results shown in Table 14 shows that this reduction is comparable to the reductions found in solving randomly-generated problems of 100 blocks (a very high degree of natural structure) presented in natural order.

Using Howard's algorithm with Natural Decomposition VDO's, the problem was solved in 52.6 seconds of total execution time, or 10.46 seconds per VDO. This is approximately a 46 percent reduction in time compared to that of the standard method. Comparison with the results shown in Table 14 shows that this reduction, again, is comparable to the reductions found in solving randomly-generated problems of 50 to 100 blocks (a high degree of natural structure) presented in natural order.

Using White's algorithm with Forecast Acceleration, the total execution time was 77.4 seconds. This is approximately a 20 per cent reduction in time compared to that of the standard method. Comparison with the results shown in Table 14 is complicated by the fact that in White's method there are no VDO's as such, in that the policy can change before the values converge. Since there were 6 policy improvements, the total time can be considered as about 13 seconds per policy, consistant again with the results for solving randomly-generated problems of 50 to 100 blocks (a high degree of natural structure).

In summary, all three of the proposed new methods are useful in reducing computation time for a highly-structured 100-state inventory problem.

It should be noted that Arbitrary Decomposition can not be ex-

pected to be useful for problems with substantially less structure than that of the inventory problem, and should be considered only as a last resort in storage-limited situations.

Successive overrelaxation (see Chapter IV) has not been tested, because of its dependence on an experimentally-determined overrelaxation factor. However, Rosenthal [99] has used successive overrelaxation with an arbitrarily-chosen overrelaxation factor on some very large, highly-structured problems in location analysis with promising results. The relative efficacies of Forecast Acceleration and successive overrelaxation remain to be determined.

CHAPTER VIII

CONCLUSIONS AND RECOMMENDATIONS

The work reported in this thesis has touched on many areas of MDP, with emphasis on large-scale computations. Some contributions have been made in cost modeling, statistical inference, state-change modeling and unifying of solution procedures. The major contributions, however, are the two decomposition methods--Arbitrary Decomposition and Natural Decomposition--and the 2-V Forecast Acceleration method. The decomposition methods have been shown to converge and have been shown to be more efficient than standard methods when applied to a wide variety of large-scale MDP problems.

The 2-V Forecast Acceleration method is a relatively efficient version of successive approximation, but has not been tested against successive overrelaxation, a standard method in numerical analysis that has been applied once to a large-scale MDP problem [99] with very encouraging results. Probably some combination of various acceleration techniques, not yet developed, would perform best. In its present form, 2-V Forecast Acceleration does not have a well developed stopping rule, cannot be combined with other acceleration techniques, and requires extra storage compared to other successive approximation techniques; thus it cannot be considered well developed.

Other new methods have been advanced herein but not developed in sufficient detail to warrant immediate use. The 3-V Forecast-Acceleration method and the two-β method are interesting ideas whose details

have yet to be worked out. It does seem clear that 3-V Forecast Acceleration is more costly and less accurate than 2-V Forecast Acceleration, and the inconclusive results of Zaldivar and Hodgson with a similar 3-point acceleration for the undiscounted case corroborate this. With respect to the two-β method, the large-scale computational experience of Hitchcock and MacQueen is encouraging in that they obtain much faster convergence at low discount factors for iterative method.

Recommended Large Scale Solution Strategies

The computational experience reported herein identifies Natural Decomposition into subproblems as the best method for MDP problems with appreciable structure. The Natural Decomposition algorithm reported in Chapter VI quickly identifies the order in which solution can proceed through successive solution of subproblems of the least possible size. The computer implementation SMDP was able to reduce total computation time in the 100-state inventory problem and in all randomly-generated test problems except those that were completely ergodic; a similar implementation was the key to Rosenthal's successful solution of location-analysis problems with more than 1000 states [99].

Since matrix inversion uses computer time proportional to N^3 for large N, and iterative techniques seem to use computer time proportional to N^2 for large N, it is widely assumed that iterative techniques such as successive approximations are indicated for truly large problems, even if larger and larger matrix inversion methods are developed. However, both arbitrary and natural decomposition schemes are iterative in the sense of trading off size for time, and it is

expected that these methods will use time more closely proportional to N^2 than to N^3 as problem sizes increase above those treated in this thesis. Another relevant factor in comparison is the fact that successive approximations is *hindered* by problem structure. In the work of Hitchcock and MacQueen [53], for example, the highly structured problems (their type III and IV matrices) take ten times as much computer time to solve as do the more artificial unstructured problems.

The solution strategy for a given large-scale problem should depend on the relative costs or availabilities of computer time and storage. For example, if storage is very limited and time is unlimited, successive approximation should be used. With the decomposition methods given here, new options are available to the programmer. The main size considerations are

1. The size of the largest matrix that can be stored in core

and 2. The size of the largest matrix that can conveniently be inverted.

Arbitrary Decomposition should be used to split the problem into roughly equal subproblems to provide size reduction for either storage or inversion; it should be used only as necessary. Natural decomposition should be used for every VDO, except in the rare case where a problem truly has no structure. The boundaries between decomposition methods and successive approximation methods are imprecise, but it is clear that successive approximation is superior only if the storage requirement for the N×N matrix (or its non-zero elements) exceeds available storage by orders of magnitude, or where time is very cheap in comparison to storage.

Recommendations for Further Research

An obvious area for further research is to develop more sophisticated rules for jumping to forecasts in the forecast acceleration procedures, and to develop ways of combining forecast acceleration with other procedures for accelerating successive approximation. In the 2-V Forecast Acceleration procedure, the crude jumping criterion used in this thesis was to jump whenever two successive forecasts were within 15 percent of each other. In every problem tested, large or small, it was noted that *iteractive* jumps, where jumping could be controlled by the ad-hoc judgement of an analyst observing the results of each iteration, would have performed far better than the rules used.

In order to derive a reasonable jumping criterion, it would make sense to study the characteristics of the time series of forecasts. This series has interesting properties; after a policy change or a jump its variance is high, and then the variance decreases asymptotically towards zero. A criterion to jump after the sample variance (or mean absolute deviation) of the forecast has decreased to within a given range would seem to be indicated. Selection of the range could presumably be done by considering the relative expected costs of false jumps and failures to jump (both of which frequently occur using the crude criterion used in the experiments in Chapter VII).

The Natural Decomposition procedure developed herein should be implemented in a version that would process non-zero elements only. This would greatly extend the size limit of its use for large problems with sparse transition matrices.

Another need uncovered by this research is for guidelines for

selection among MDP solution techniques. This thesis covers all standard methods and some new ones, yet no comprehensive formal guidelines were developed. It is difficult to tell from the literature, including this thesis, what method is likely to be most efficient in solving a given MDP problem. Any formal guidelines would need to be based on extensive empirical computational experience, far in excess of that reported here.

APPENDIX I

COMPUTER PROGRAM MDP AND AN EXAMPLE RUN

```
00310     REM***  DATA INPUT  ***
00311    PRINT'INPUT THE DISCOUNT FACTOR';
00312    INPUT B
00313    IF B>0 AND B<1 THEN GO TO 316
00314    PRINT'YOUR DISCOUNT FACTOR MUST BE BETWEEN 0 AND 1;
00315    GO TO 312                                        TRY AGAIN.'
00316    PRINT'YOUR DISCOUNT FACTOR IS EQUIVALENT TO'100*
NT'                                                  (1/B-1);'PER CE
00317    PRINT'INTEREST PER TRANSITION PERIOD.'
00320    PRINT'INPUT THE NUMBER OF STATES';
00330    INPUT N
00340    PRINT
00360     MAT K=ZER(N)
00370    DIM P(10,10,10),C(10,10,10)
00375     FOR I=1 TO N
00390    PRINT
00400    PRINT'HOW MANY ACTIONS AT STATE'I;
00410     INPUT K(I)
00415     FOR K=1 TO K(I)
00430    PRINT'GIVE ROW';I;'OF  P  UNDER ACTION';K
00440    MAT Q=ZER(N)
00450    Q0=0
00460    MAT INPUT Q
00470    IF NUM=N GO TO 500
00480    PRINT'NOT ENOUGH DATA; TRY AGAIN.'
00490    GO TO 440
00500    FOR J=1 TO N
00510    IF Q(J)>=0 GO TO 540
00520    PRINT'P('I;','J;')  MUST BE NON-NEGATIVE; TRY AGAIN'
00530    GO TO 440
00540    Q0=Q0+Q(J)
00550    NEXT J
00560    IF ABS(Q0-1)<.000001 GO TO 590
00570    PRINT'THE SUM OF THE ELEMENTS OF ROW'I;'MUST BE ONE;
'                                                    TRY AGAIN.
00580    GO TO 440
00590    FOR J=1 TO N
00600    P(I,J,K)=Q(J)
00610    NEXT J
00630    PRINT'GIVE ROW';I;'OF  C  UNDER ACTION';K
00640    MAT Q=ZER(N)
```

```
00650     MAT INPUT Q
00660     IF NUM=N GO TO 690
00670     PRINT'NOT ENOUGH DATA; TRY AGAIN.'
00680     GO TO 640
00690     FOR J=1 TO N
00700     C(I,J,K)=Q(J)
00710     NEXT J
00730      NEXT K
00740      NEXT I
00741      REM***  EVALUATION OF THE MATRIX OF IMMEDIATE
00742      MAT R=ZER(N,1O)                        REWARDS  R  ***
00743      FOR I=1 TO N
00744      FOR K=1 TO K(I)
00745      FOR J=1 TO N
00746      R(I,K)=R(I,K)+C(I,J,K)*P(I,J,K)
00747      NEXT J
00748      NEXT K
00749      NEXT I
00750      REM***  END OF THE EVALUATION OF  R  ***
00751      REM
00752      REM
00755      REM***  DIMENSIONING OF REQUIRED MATRICES  ***
00760      REM
00765      MAT A=ZER(N,1)
00770      MAT F=ZER(N,N)
00775      MAT E=IDN(N,N)
00780      MAT G=ZER(N,N)
00785      MAT H=ZER(N,N)
00790      MAT V=ZER(N,1)
00795      MAT W=ZER(N,1)
00800      MAT D=ZER(N)
00805      REM
00810      REM***  END OF DIMENSIONING OF MATRICES  ***
00815      REM
00820      REM
00825      REM*** CHOICE OF PROGRAM OPTION  ***
00830      REM
00835      PRINT'WHICH OPTION DO YOU WANT';
00840      INPUT O$
00845      REM
00850      REM
00855      REM*** EXECUTION  ***
00900      REM
00905      REM***  OPTION ONE  ***
00910      REM
00915      IF O$<>'OPTION ONE' GO TO 1000
00920      GOSUB 4000
00925      GOSUB 6000
00930      MAT W=V
00935      GOSUB 7000
00940      MAT Q=D
```

```
00945     PRINT'ANOTHER ITERATION';
00950     INPUT B$
00955     IF B$='NO' THEN STOP
00960     GO TO 925
00965     REM
00970     REM
01000     REM***  OPTION TWO  ***
01005     REM
01010     IF O$<>'OPTION TWO' GO TO 1500
01015     G$='GAUSS/SEIDEL'
01020     GOSUB 4000
01025     GOSUB 6000
01030     MAT W=V
01035     GOSUB 7000
01040     MAT Q=D
01045     PRINT'ANOTHER ITERATION';
01050     INPUT B$
01055     IF B$='NO' THEN STOP
01060     GO TO 1025
01065     REM
01070     REM
01500     REM***  OPTION THREE  ***
01505     REM
01510     IF O$<>'OPTION THREE' GO TO 2000
01520     GOSUB 5000
01530    GOSUB 7000
01540    MATW=V
01550    PRINT'ANOTHER  PIR';
01560    INPUT B$
01570    IF B$='NO' THEN STOP
01580    GO TO 1530
01590    REM
01600    REM
02000    REM***  OPTION FOUR  ***
02010    REM
02020    IF O$<>'OPTION FOUR' GO TO 2500
02030    G$='GAUSS/SEIDEL'
02040    GOSUB 5000
02050    GOSUB 7000
02060    PRINT'ANOTHER  PIR';
02070    INPUT B$
02080    IF B$='NO' THEN STOP
02090    GO TO 2050
02100    REM
02110    REM
02500    REM***  OPTION FIVE  ***
02510    REM
02520    IFO$<>'OPTION FIVE' GO TO 3000
02530    G$='GAUSS/SEIDEL'
02540    GOSUB 5000
02550    GOSUB 7000
```

```
02560     PRINT'ANOTHER  PIR';
02570     INPUT B$
02580   IF B$='YES' GO TO 2550
02590     MAT Q=D
02600     GOSUB 6000
02610     MAT W=V
02620     GOSUB 7000
02630     MAT Q=D
02640     PRINT'ANOTHER  PIR';
02650     INPUT B$
02660     IF B$='NO' THEN STOP
02670     GO TO 2550
02680     REM
02690     REM
03000     REM***  OPITON SIX  ***
03010     REM
03020     G$='GAUSS/SEIDEL'
03030     GOSUB 5000
03040     GOSUB 7000
03050     GOSUB 7000
03060     MAT Q=D
03070     GOSUB 6000
03080     MAT W=V
03090     PRINT'ANOTHER ITERATION';
03100     INPUT B$
03110     IF B$='NO' THEN STOP
03120     GO TO 3040
03130     REM
03140     REM
04000     REM***  INITIALIZATION FOR HOWARD'S METHOD  ***
04010     REM
04020     PRINT'WHAT IS YOUR INITIAL POLICY'
04030     MAT INPUT Q
04040     PRINT
04050     RETURN
04060     REM
04070     REM
05000     REM***  INITIALIZATION FOR THE METHOD OF SUCCESSIVE
ONS  ***                                                  APPROXIMATI
05010     PRINT'GIVE THE INITIAL VALUES:'
05020     FOR I=1 TO N
05030     PRINT'V('I;')=';
05040    INPUT W(I,1)
05050     NEXT I
05060     PRINT
05070     RETURN
05080     REM
05090     REM
06000     REM***  START OF  VDO  ***
06010     REM
06020     FOR I=1 TO N
```

```
06030     FOR J=1 TO N
06040     K=Q(I)
06050     A(I,1)=R(I,K)
06060     F(I,J)=P(I,J,K)
06070     NEXT J
06080     NEXT I
06090     MAT F=(B)*F
06100     MAT G=E-F
06110     MAT H=INV(G)
06120     MAT V=H*A
06130     PRINT'THE CURRENT  VDO  GIVES:'
06140     MAT PRINT V
06150     PRINT
06160     PRINT
06170     RETURN
06180     REM***  END OF  VDO   ***
06190     REM
06200     REM
07000     REM***  START OF  PIR  ***
07010     REM
07020     PRINT'POLICY IMPROVEMENT ROUTINE'
07030     FOR I=1 TO N
07040     PRINT
07050     PRINT'FOR STATE'I;':'
07060     MAT L=ZER(K(I))
07070     FOR K=1 TO K(I)
07080     FOR J=1 TO N
07090     L(K)=L(K)+P(I,J,K)*W(J,1)
07100     NEXT J
07110     L(K)=R(I,K)+B*L(K)
07120     PRINT'ACTION'K;'GIVES  V('I;')='L(K)
07130     NEXT K
07140     L0=L(1)
07150     K0=1
07160     FOR K=1 TO K(I)
07170     IF L0>L(K) GO TO 7200
07180     L0=L(K)
07190     K0=K
07200     NEXT K
07210     V(I,1)=L0
07220     D(I)=K0
07230     PRINT'THUS, V('I;')='V(I,1),'D('I;')='D(I)
07240     IF G$='GAUSS/SEIDEL' THEN W(I,1)=V(I,1)
07250     NEXT I
07260     PRINT
07270     PRINT
07280     RETURN
07290     REM***  END OF  PIR  ***
07300     END
```

(Example Run)

```
WHICH ØPTIØN DØ YØU WANT? 2,2ØPTIØN TWØ
WHAT IS YØUR INITIAL PØLICY
? 2,2

THE CURRENT  VDØ  GIVES:

 27.427484
 53.518278


PØLICY IMPRØVEMENT RØUTINE

FØR STATE 1 :
ACTIØN 1 GIVES  V( 1 )= 29.925592
ACTIØN 2 GIVES  V( 1 )= 27.427484
THUS, V( 1 )= 29.925592        D( 1 )= 1

FØR STATE 2 :
ACTIØN 1 GIVES  V( 2 )= 52.108095
ACTIØN 2 GIVES  V( 2 )= 53.743107
THUS, V( 2 )= 53.743107        D( 2 )= 2


ANØTHER ITERATIØN? YES
THE CURRENT  VDØ  GIVES:

 34.843743
 57.031241


PØLICY IMPRØVEMENT RØUTINE

FØR STATE 1 :
ACTIØN 1 GIVES  V( 1 )= 34.843742
ACTIØN 2 GIVES  V( 1 )= 31.74843
THUS, V( 1 )= 34.843742        D( 1 )= 1

FØR STATE 2 :
ACTIØN 1 GIVES  V( 2 )= 55.58593
ACTIØN 2 GIVES  V( 2 )= 57.031241
THUS, V( 2 )= 57.031241        D( 2 )= 2


ANØTHER ITERATIØN? NØ
```

APPENDIX II

SVDO (SPARSE VDO) COMPUTER CODE

```
      SUBROUTINE SVDO(N,BETA,R,P,V,
     +B,IC,ISET)
      REAL P(100,100),V(100),R(100)
      LOGICAL B(100,100)
      INTEGER IC(100),ISET(100)
C*****INITIALIZE
      IDENSE=0
      ICOUNT=0
      DO 10 I=1,N
      DO 20 J=1,N
      B(I,J)=(P(I,J) .GT. 0.0)
      IF (.NOT. B(I,J)) GO TO 20
      IDENSE=IDENSE+1
20    CONTINUE
      B(I,I)=.FALSE.
10    ISET(I)=I
C     CHECK FOR ABSORBING STATES
      DO 40 I=1,N
      IF (P(I,I) .LT. 1.0) GO TO 40
      B(I,I)=.TRUE.
      ICOUNT=ICOUNT+1
40    CONTINUE
C
C****ALL STATEMENTS ABOVE THIS LINE ARE EXECUTED ONLY ONCE*
C     EXECUTION
30    CALL ZEROW(N,B,IZEROW)
      IF (IZEROW .GT. 0) GO TO 51
    3 FORMAT (' 3')
      CALL CYCLE(N,B,ISET,IZEROW,IC)
51    CALL SOLVE(N,IZEROW,P,V,R,BETA,ICOUNT,IC,ISET,B,
      IF (ICOUNT .LT. N) GO TO 30                IDENSE)
      RETURN
      END
```

```
      SUBROUTINE ZEROW(N,B,IZEROW)
      LOGICAL B(100,100)
C     IZEROW IS A NONDELETED STATE THAT DOES NOT
C          LEAD TO ANY OTHER NONDELETED STATE, IF IT EXISTS
C          OTHERWISE IZEROW=0.
      DO 10 I=1,N
      IF (B(I,I)) GO TO 10
      DO 20 J=1,N
      IF (B(J,J) .OR. .NOT. B(I,J)) GO TO 20
      GO TO 10
20    CONTINUE
      IZEROW=I
      RETURN
10    CONTINUE
      IZEROW=0
      RETURN
      END

      SUBROUTINE CYCLE (N,B,ISET,IZEROW,IC)
      LOGICAL B(100,100)
      INTEGER ISET(100),IC(100)
C     B(I,I) =.I IS DELETED.
C     ISET(I)=0 IFF I IS IN THE PATH
      NC=0
C     FIND FIRST STATE
      DO 10 J=1,N
      IF (B(J,J)) GO TO 10
      IL=J
      GO TO 30
10    CONTINUE
      GO TO 190
C     ADD STATE TO PATH
30    NC=NC+1
      IC(NC)=IL
      ISET(IL)=0
C     FIND ANOTHER STATE: IF NOT A REPEAT ADD TO PATH
      DO 40 J=1,N
      IF (B(J,J) .OR. .NOT. B(IL,J)) GO TO 40
      IL=J
      GO TO 50
40    CONTINUE
      GO TO 190
50    IF (ISET(IL).NE. 0) GO TO 30
C     REPEAT FOUND. FIND FIRST OCURRENCE
      IR=IL
      DO 70 I=1,NC
      IF (IC(I) .NE. IR) GO TO 70
      NR=I
      GO TO 80
70    CONTINUE
```

```
C     CYCLE IDENTIFIED  IR=IC(NR),IC(NR+1),...,IC(NC),IR
C     COLLAPSE CYCLE
80    B(IR,IR)=.TRUE.
      NR1=NR+1
      DO 100 I=NR1,NC
      JR=IC(I)
      B(JR,JR)=.TRUE.
      ISET(JR)=IR
      DO 110 K=1,N
      IF (ISET(K) .EQ. JR) ISET(K)=IR
      IF (B(K,K)) GO TO 110
      B(IR,K)=B(IR,K) .OR. B(JR,K)
      B(K,IR)=B(K,IR) .OR. B(K,JR)
110   CONTINUE
100   CONTINUE
C     CHECK ROW(IR) FOR ZEROW
      DO 120 J=1,N
      IF (B(J,J) .OR. .NOT. B(IR,J)) GO TO 120
      IL=J
      B(IR,IR)=.FALSE.
      NC=NR
      GO TO 50
120   CONTINUE
      DO 140 I=1,NR
      JR=IC(I)
140   ISET(JR)=JR
      IZEROW=IR
      RETURN
190   WRITE(6,191)
191   FORMAT (22H ERROR EXIT FROM CYCLE)
      STOP
      END


      SUBROUTINE SOLVE(N,IZEROW,P,V,R,BETA,ICOUNT,IC,ISET,B,
     +IDENSE)
      LOGICAL B(100,100)
      INTEGER IC(100),ISET(100)
      REAL P(100,100),V(100),R(100)
      DIMENSION A(100,101),IGARB1(100),GARBG2(1)
      IZ=IZEROW
C     DETERMINE SIZE OF SYSTEM & IDENTIFY VARIABLES
      ISIZE=0
      DO 10 I=1,N
      IF (ISET(I) .NE. IZ) GO TO 10
      ISIZE=ISIZE+1
      B(I,I)=.FALSE.
      IC(ISIZE)=I
10    CONTINUE
    1 FORMAT (' 7', 515)
```

```
      IF (ISIZE .GT. 1) GO TO 30
      SUM=0.0
      DO 20 I=1,N
20    IF (B(IZ,I)) SUM=SUM+P(IZ,I)*V(I)
      V(IZ)=(R(IZ)+BETA*SUM)/(1-BETA*P(IZ,IZ))
      B(IZ,IZ)=.TRUE.
      ITER=0
      GO TO 190
C     UPDATE RIGHT-HAND-SIDE
30    DO 50 I=1,ISIZE
      JR=IC(I)
      SUM=0.0
      DO 40 K=1,N
      IF (.NOT. B(JR,K)) GO TO 40
      IF (ISET(K) .EQ. IZ) GO TO 40
      SUM=SUM+P(JR,K)*V(K)
40    CONTINUE
      R(JR)=R(JR)+BETA*SUM
      B(JR,JR)=.TRUE.
50    CONTINUE
C*** SET-UP A MATRIX  A  FOR GJR ***
      ISIZE1=ISIZE+1
      DO 80 I=1,ISIZE
      IR=IC(I)
      DO 110 J=1,ISIZE
      JR=IC(J)
      A(I,J)=-BETA*P(IR,JR)
      IF(I.EQ.J) A(I,J)=1.+A(I,J)
110   CONTINUE
80    A(I,ISIZE1)=R(IR)
C
C
      GARBG2(1)=4.
      NR=100
      NC=101
      CALL GJR(A,NC,NR,ISIZE,ISIZE1,$1900,IGARB1,GARBG2)
      DO 100 I=1,ISIZE
      IR=IC(I)
100   V(IR)=A(I,ISIZE1)
C
C
190      ICOUNT=ICOUNT+ISIZE
      RETURN
1900  WRITE(6,4444)
4444  FORMAT('0JO:  OVERFLOW')
      STOP
      END
```

APPENDIX III

COMPUTER PROGRAM FOR ARBITRARY DECOMPOSITION (DMDP)

```
00160  V3=0
00170  O$='OPTION ONE'
00310       REM*** DATA INPUT ***
00311      PRINT'INPUT THE DISCOUNT FACTOR';
00312      INPUT B
00313      IF B>0 AND B<1 THEN GO TO 3'6
00314      PRINT'YOUR DISCOUNT FACTOR MUST BE BETWEEN 0 AND
00315      GO TO 312                                    1; TRY AGAIN.'
00316      PRINT'YOUR DISCOUNT FACTOR IS EQUIVALENT TO'100*
           (1/B-1);'PER CENT'
00317      PRINT'INTEREST PER TRANSITION PERIOD.'
00320      PRINT'INPUT THE NUMBER OF STATES';
00330      INPUT N
00340      PRINT
00360       MAT K=ZER(N)
00370      DIM P(10,10,10),C(10,10,10)
00375       FOR I=1 TO N
00390      PRINT
00400      PRINT'HOW MANY ACTIONS AT STATE'I;
00410       INPUT K(I)
00415       FOR K=1 TO K(I)
00440      MAT Q=ZER(N)
00450      Q0=0
00460      MAT INPUT Q
00470      IF NUM=N GO TO 500
00480      PRINT'NOT ENOUGH DATA; TRY AGAIN.'
00490      GO TO 440
00500      FOR J=1 TO N
00510      IF Q(J)>=0 GO TO 540
00520      PRINT'P('I;','J;')  MUST BE NON-NEGATIVE; TRY
00530      GO TO 440                                    AGAIN'
00540      Q0=Q0+Q(J)
00550      NEXT J
00560      IF ABS(Q0-1)<.000001 GO TO 590
00570      PRINT'THE SUM OF THE ELEMENTS OF ROW'I;'MUST BE
00580      GO TO 440                                    ONE; TRY AGAIN.'
00590      FOR J=1 TO N
00600      P(I,J,K)=Q(J)
00610      NEXT J
00640      MAT Q=ZER(N)
00650      MAT INPUT Q
00660      IF NUM=N GO TO 690
```

```
00670       PRINT'NOT ENOUGH DATA; TRY AGAIN.'
00680       GO TO 640
00690       FOR J=1 TO N
00700       C(I,J,K)=0(J)
00710       NEXT J
00730        NEXT K
00740        NEXT I
00741  N=3
00742        MAT R=ZER(N,10)
00743        FOR I=1 TO N
00744        FOR K=1 TO K(I)
00745        FOR J=1 TO N
00746        R(I,K)=R(I,K)+C(I,J,K)*P(I,J,K)
00747        NEXT J
00748        NEXT K
00749        NEXT I
00751  PRINT'THE UPDATED VALUES ARE:'
00752   FOR I=1 TO 2
00753   FOR K=1 TO K(I)
00754   R(I,K)=P(I,K)+B*P(I,3,K)*V3
00755   PRINT'P('I;',';'K;')='P(I,K)
00756   NEXT K
00757   NEXT I
00758   N=2
00760        REM
00765        MAT A=ZER(N,1)
00770        MAT F=ZER(N,N)
00775        MAT E=IDN(N,N)
00780        MAT G=ZER(N,N)
00785        MAT H=ZER(N,N)
00790        MAT V=ZER(N,1)
00795        MAT W=ZER(N,1)
00800        MAT D=ZER(N)
00801   MAT B=ZER(N)
00805        REM
00810        REM***  END OF DIMENSIONING OF MATRICES  ***
00815        REM
00820        REM
00825        REM*** CHOICE OF PROGRAM OPTION  ***
00830        REM
00845        REM
00850        REM
00855        REM*** EXECUTION  ***
00900        REM
00905        REM***  OPTION ONE  ***
00910        REM
00915        IF O$<>'OPTION ONE' GO TO 1000
00920        GOSUB 4000
00925        GOSUB 6000
00930        MAT W=V
00935        GOSUB 7000
```

```
00940   MAT R=D
00945        PRINT'ANOTHER ITERATION';
00950        INPUT B$
00955   IF B$='NO' GO TO 961
00960        GO TO 925
00961   FOR K=1 TO K(3)
00962   V3=(R(3,K)+B*P(3,1,K)*V(1,1)+B*P(3,2,K)*V(2,1))/
00963   PRINT'FOR ACTION 'K;'V(3)='V3          (1-B*P(3,3,K))
00964   NEXT K
00965   PRINT'ENTER V(3)';
00966   INPUT V3
00967   GO TO 741
00970        REM
01000        REM***  OPTION TWO  ***
01005        REM
01010        IF O$<>'OPTION TWO' GO TO 1500
01015        G$='GAUSS/SEIDEL'
01020        GOSUB 4000
01025        GOSUB 6000
01030        MAT W=V
01035        GOSUB 7000
01040        MAT Q=D
01045        PRINT'ANOTHER ITERATION';
01050        INPUT B$
01055        IF B$='NO' THEN STOP
01060        GO TO 1025
01065        REM
01070        REM
01500        REM***  OPTION THREE  ***
01505        REM
01510        IF O$<>'OPTION THREE' GO TO 2000
01520        GOSUB 5000
01530       GOSUB 7000
01540       MATW=V
01550       PRINT'ANOTHER  PIP';
01560       INPUT B$
01570       IF B$='NO' THEN STOP
01580       GO TO 1530
01590       REM
01600       REM
02000       REM***  OPTION FOUR  ***
02010       REM
02020       IF O$<>'OPTION FOUR' GO TO 2500
02030       G$='GAUSS/SEIDEL'
02040       GOSUB 5000
02050       GOSUB 7000
02060       PRINT'ANOTHER  PIR';
02070       INPUT B$
02080       IF B$='NO' THEN STOP
02090       GO TO 2050
02100       REM
```

```
02110       REM
02500       REM***  OPTION FIVE  ***
02510       REM
02520       IF G$<>'OPTION FIVE' GO TO 3000
02530       G$='GAUSS/SEIDEL'
02540       GOSUB 5000
02550       GOSUB 7000
02560       PRINT'ANOTHER  PIP';
02570       INPUT B$
02580     IF B$='YES' GO TO 2550
02590       MAT G=D
02600       GOSUB 6000
02610       MAT W=V
02620       GOSUB 7000
02630       MAT G=D
02640       PRINT'ANOTHER  PIP';
02650       INPUT B$
02660       IF B$='NO' THEN STOP
02670       GO TO 2550
02680       REM
02690       REM
03000       REM***  OPITON SIX  ***
03010       REM
03020       G$='GAUSS/SEIDEL'
03030       GOSUB 5000
03040       GOSUB 7000
03050       GOSUB 7000
03060       MAT G=D
03070       GOSUB 6000
03080       MAT W=V
03090       PRINT'ANOTHER ITERATION';
03100       INPUT B$
03110       IF B$='NO' THEN STOP
03120       GO TO 3040
03130       REM
03140       REM
04000       REM***  INITIALIZATION FOR HOWARD'S METHOD  ***
04010       REM
04020       PRINT'WHAT IS YOUR INITIAL POLICY'
04030    MAT INPUT B
04040       PRINT
04050       RETURN
04060       REM
04070       REM
05000       REM***  INITIALIZATION FOR THE METHOD OF SUCCESSIVE
05010       PRINT'GIVE THE INITIAL VALUES:'          APPROXIMATIONS
05020       FOR I=1 TO N
05030       PRINT'V('I;')=';
05040      INPUT W(I,1)
05050       NEXT I
05060       PRINT
```

```
05070       RETURN
05080       REM
05090       REM
06000       REM***  START OF  VDO  ***
06010       REM
06020       FOR I=1 TO N
06030       FOR J=1 TO N
06040   K=P(I)
06050       A(I,1)=R(I,K)
06060       F(I,J)=P(I,J,K)
06070       NEXT J
06080       NEXT I
06090       MAT E=(B)*F
06100       MAT G=E-F
06110       MAT H=INV(G)
06120       MAT V=H*A
06130       PRINT'THE CURRENT  VDO  GIVES:'
06140       MAT PRINT V
06150       PRINT
06160       PRINT
06170       RETURN
06180       REM***  END OF  VDO   ***
06190       REM
06200       REM
07000       REM***  START OF  PIR  ***
07010       REM
07020       PRINT'POLICY IMPROVEMENT ROUTINE'
07030       FOR I=1 TO N
07040       PRINT
07050       PRINT'FOR STATE'I;':'
07060       MAT L=ZER(K(I))
07070       FOR K=1 TO K(I)
07080       FOR J=1 TO N
07090       L(K)=L(K)+P(I,J,K)*W(J,1)
07100       NEXT J
07110       L(K)=R(I,K)+B*L(K)
07120       PRINT'ACTION'K;'GIVES  V('I;')='L(K)
07130       NEXT K
07140       L0=L(1)
07150       K0=1
07160       FOR K=1 TO K(I)
07170       IF L0>L(K) GO TO 7200
07180       L0=L(K)
07190       K0=K
07200       NEXT K
07210       V(I,1)=L0
07220       D(I)=K0
07230       PRINT'THUS, V('I;')='V(I,1),'D('I;')='D(I)
07240       IF G$='GAUSS/SEIDEL' THEN W(I,1)=V(I,1)
07250       NEXT I
```

```
07260       PRINT
07270       PRINT
07280       RETURN
07290       REM***  END OF  PIR  ***
07300       END
```

APPENDIX IV

FORTRAN CODE FOR NATURAL DECOMPOSITION (SMDP)

```
C      PROGRAM  SMDP
C      THIS PROGRAM SOLVES A MARKOV DECISION PROCESS PROBLEM
E      WITH A SPARSE
C  .   TRANSITION PROBABILITY MATRIX.
C
C
       DIMENSION P(50,50,10),PK(50,50),C(50),R(50,10),RK(50)
       DIMENSION KMAX(50),ALMAX(50),KA(50),KOLD(50),V(50)
C
C
C*** INPUT OF  P, BETA, N  ***
C
       READ(5,1) BETA,N
1      FORMAT(F10.0,I3)
       DO 100 I=1,N
       READ(5,2)KA(I)
2      FORMAT(I3)
       KK=KA(I)
       DO 200 K=1,KK
       READ(5,3)(P(I,J,K),J=1,N)
3      FORMAT(8F10.8)
       READ(5,4)(C(J),J=1,N)
4      FORMAT(8F10.4)
C
C*** EVALUATION OF THE MATRIX OF IMMEDIATE REWARDS  R  ***
 C
       R(I,K)=0.
       DO 300 J=1,N
300    R(I,K)=R(I,K)+P(I,J,K)*C(J)
200    CONTINUE
100    CONTINUE
C
C
C
C*** INITIALIZATION ***
C
       KOUNT=0
       DO 400 I=1,N
       KOLD(I)=-1
400    V(I)=0.
C
C
```

```
C*** BEGIN ITERATIONS ***
C
600    KOUNT=KOUNT+1
       CALL PIR(BETA,N,P,V,R,KMAX,ALMAX,KA)
C
C
C*** STOPPING RULES ***
C
       DO 700 I=1,N
       IF(KMAX(I).NE.KOLD(I)) GO TO 800
       IF(ABS(V(I)-ALMAX(I)).GT.0.1) GO TO 800
700    CONTINUE
C
C
C*** OUTPUT ***
C
       WRITE(6,5)KOUNT,((I,KMAX(I),V(I)),I=1,N)
5      FORMAT('1 NUMBER OF ITERATIONS=',I5//'0STATE  ACTION
      $(I6,I8,F10.4))                                  VALUE'//
       STOP
C
C
C*** CONTINUE ITERATIONS ***
C
800    DO 900 I=1,N
       V(I)=ALMAX(I)
900    KOLD(I)=KMAX(I)
       IDENSE=0
       DO 500 I=1,N
       KK=KMAX(I)
       RK(I)=R(I,KK)
       DO 500 J=1,N
       PK(I,J)=P(I,J,KK)
       IF(PK(I,J).GT.0)IDENSE=IDENSE+1
500    CONTINUE
           CALL GJVDO(PK,RK,BETA,N,V)
       GO TO 600
       END
```

```
      SUBROUTINE GJVDO(P,R,BETA,N,V)
      DIMENSION P(100,100),R(100),A(100,101),IGARB1(100)
      DIMENSION V(100),GARB2(1)
      N1=N+1
      DO 10 I=1,N
      DO 20 J=1,N
      A(I,J)=-BETA*P(I,J)
      IF(I.EQ.J)A(I,J)=1.+A(I,J)
20    CONTINUE
      A(I,N1)=R(I)
10    CONTINUE
      GARB2(1)=4.
      NC=101
      NR=100
      CALL GJR(A,NC,NR,N,N1,$1000,IGARB1,GARB2)
      DO 30 I=1,N
30    V(I)=A(I,N1)
      RETURN
100   WRID@(6,1)
3K    @ FRMAT(' 0J0:  0ERFLOW')
      STOP
      END
```

APPENDIX V

RANDOM TEST PROBLEM GENERATION PROCEDURE

Hitchcock and MacQueen [53] provided a method of producing randomly generated test problems of several types of transition matrix. In their type 1 matrix, all row entries were generated from a uniform distribution and normalized. In their type 2 matrix, a large probability $p(i,1)$ was entered in the left-hand column for each row i, and the remaining entries were uniformly distributed. In their type 3 matrix, a large probability was provided on the main diagonal, $p(i,i)$ and the remaining entries were uniformly distributed. These matrices are highly unrealistic, and in a medium to large problem nearly all probabilities tend to $1/N$, where N is the number of states.

Their type 4 matrix had large probabilities scattered according to a set pattern. Even these matrices seemed to have less structure than those of any application problem encountered by this author.

A good generator should generate Markov transition matrices with a given structure, so that the user can produce test problems similar to the actual problems the test problems are substituting for.

The generator used here requires the user to input the number of equivalence classes with recurrent states, the number of equivalence classes with transient states, and the number of states in each class. It then generates submatrices for each equivalence class and assembles them. The number of non-zero elements within each submatrix can be set

or generated randomly. Thus the user controls the structure. The program also generates state-occupancy rewards according to a specified distribution. The rows and columns can be scrambled if desired.

The following example run generates a scrambled matrix where the user had previously specified (within the program) that there were to be no zero elements withing the dense portions of the matrix and that the rewards were to be distributed uniformly.

```
RAND-GEN      22:22:04     8 AUG 75

HOW MANY EQUIVALENCE CLASSES WITH RECURRENT STATES? 2
HOW MANY EQUIVALENCE CLASSES WITH TRANSIENT STATES? 3
HOW MANY STATES IN EQUIVALENCE CLASS NUMBER 1 ?2
HOW MANY STATES IN EQUIVALENCE CLASS NUMBER 2 ?3
HOW MANY STATES IN EQUIVALENCE CLASS NUMBER 3 ?2
HOW MANY STATES IN EQUIVALENCE CLASS NUMBER 4 ?3
HOW MANY STATES IN EQUIVALENCE CLASS NUMBER 5 ?2

THE TRANSITION MATRIX GENERATED IS:

 .590  .4100.0000.0000.0000.0000.0000.0000.0000.0000.0000.000
 .388  .6120.0000.0000.0000.0000.0000.0000.0000.0000.0000.000
0.0000.000  .149  .420  .4310.0000.0000.0000.0000.0000.0000.000
0.0000.000  .012  .176  .8120.0000.0000.0000.0000.0000.0000.000
0.0000.000  .188  .226  .5860.0000.0000.0000.0000.0000.0000.000
 .065  .193  .166  .217  .128  .138  .0910.0000.0000.0000.0000.000
 .043  .175  .069  .090  .213  .267  .1430.0000.0000.0000.0000.000
 .158  .046  .017  .187  .167  .106  .066  .077  .077  .0980.0000.000
 .069  .142  .012  .175  .136  .119  .113  .092  .078  .0640.0000.000
 .067  .072  .093  .148  .142  .091  .043  .129  .142  .0730.0000.000
 .129  .100  .127  .029  .125  .100  .043  .038  .093  .082  .062  .072
 .113  .029  .125  .137  .009  .117  .135  .030  .029  .119  .032  .125
```

```
THE TRANSFORMED TRANSITION MATRIX IS:

 .5900.0000.0000.0000.000 .4100.0000.0000.0000.0000.0000.000
0.000 .176 .0120.0000.0000.0000.0000.0000.0000.000 .8120.000
0.000 .420 .1490.0000.0000.0000.0000.0000.0000.000 .4310.000
 .043 .090 .069 .143 .267 .1750.0000.0000.0000.000 .2130.000
 .065 .217 .166 .091 .138 .1930.0000.0000.0000.000 .1280.000
 .3880.0000.0000.0000.000 .6120.0000.0000.0000.0000.0000.000
 .129 .029 .127 .043 .100 .100 .062 .038 .093 .082 .125 .072
 .158 .187 .017 .066 .106 .0460.000 .077 .077 .098 .1670.000
 .069 .175 .012 .113 .119 .1420.000 .092 .078 .064 .1360.000
 .067 .148 .093 .043 .091 .0720.000 .129 .142 .073 .1420.000
0.000 .226 .1880.0000.0000.0000.0000.0000.0000.000 .5860.000
 .113 .137 .125 .135 .117 .029 .032 .030 .029 .119 .009 .125

WHERE THE STATES ARE:

 1
 4
 3
 7
 6
 2
 11
 8
 9
 10
 5
 12


WHAT IS THE SMALLEST POSSIBLE VALUE OF R
? -100
WHAT IS THE LARGEST? 100
  -89.01530
   27.19453
  -17.09549
  -23.39664
   85.49424
  -30.50089
   84.70879
  -85.01506
  -72.04966
   44.82019
    63.09869
  -16.60242


 TIME :     .481
```

BIBLIOGRAPHY

1. Antelman, GR, Russell, CB and IR Savage "Surveillance Problems: Two-Dimensional with Continuous Surveillance" SIAM Journal of Control Vol 5, 245-67 (1967)

2. Aoki, M Optimization of Stochastic Systems, Academic Press (1967)

3. Astrom, KJ "Optimal Control of Markov Processes with Incomplete State Information" Journal of Mathematical Analysis Applications Vol 10, 174-205 (1965)

4. Barfoot, CB "Markov Duels" Operations Research Vol 22, 318-30 (1974)

5. Beckmann, MJ Dynamic Programming of Economic Decisions, Springer-Verlag (1968)

6. Bellman, R "A Markovian Decision Process" Journal of Mathematical Mechanics Vol 6, 679-84 (1957)

7. Bellman, R Dynamic Programming, Princeton University Press (1957)

8. Bellman, R and D Blackwell On a Particular Non-Zero Sum Game, Rand McNally (1949)

9. Bellman, R and JP Lasalle On Non-Zero Sum Games and Stochastic Processes, Rand McNally (1949)

10. Benes, VE "Programming and Control Problems Arising from Optimal Routing in Telephone Networks" SIAM Journal of Control Vol 4, 6-18 (1966)

11. Benes, VE "Programming and Control Problems Arising from Optimal Routing in Telephone Networks" Bell System Technical Journal Vol 45, 1373-438 (1966)

12. Blackwell, D "Discrete Dynamic Programming" Annals of Mathematical Statistics Vol 33, 719-26 (1962)

13. Blackwell, D "Discounted Dynamic Programming" Annals of Mathematical Statistics Vol 36, 226-35 (1965)

14. Brown, BW "On the Iterative Method of Dynamic Programming on a Finite Space Discrete Time Markov Process" Annals of Mathematical Statistics Vol 36, 1279-85 (1965)

15. Carnahan, B, Luther, HA and JO Wilkes Applied Numerical Methods, John Wiley & Sons (1969)

16. Dantzig, GB and P Wolfe "The Decomposition Algorithm for Linear Programming" Econometrica Vol 9 (1961). Also, Operations Research Vol 8 (1960)

17. De Ghellinck, GT "Les Problémes de Décisions Séquentielles" Cahiers Centre d'Etudes de Recherche Operationelle Vol 2, 161-79 (1960)

18. De Ghellinck, GT and GD Eppen "Linear Programming Solutions for Separable Markovian Decision Problems" Management Science Vol 13, 371-94 (1967)

19. Denardo, EV "Contraction Mappings in the Theory Underlying Dynamic Programming" SIAM Review Vol 9, 165-77 (1967)

20. Denardo, EV "Separable Markovian Decision Problems" Management Science Vol 14, 451-62 (1968)

21. Denardo, EV "On Linear Programming in a Markov Decision Problem" Management Science Vol 16, 281-8 (1970)

22. Denardo, EV "Multichain Markov Renewal Programs" SIAM Journal of Applied Mathematics Vol 16, 468-87 (1968)

23. Denardo, EV and LG Mitten "Elements of Sequential Decision Processes" Journal of Industrial Engineering Vol 18, 106-11 (1967)

24. D'Epenoux, F "Sur un Probleme de Production de Stockage dans l'aleatoire" Revue Francaise de Recherche Operationelle Vol 14, 3-16 (1960). [English translation: Management Science Vol 10, 98-109 (1963)]

25. Derman, C "On Sequential Decisions and Markov Chains" Management Science Vol 9, 16-24 (1962)

26. Derman, C "On Optimal Replacement Rules When Changes of State Are Markovian" Mathematical Optimization Techniques (RE Bellman ed.) University of California Press, Berkeley (1963)

27. Derman, C "Optimal Replacement and Maintenance Under Markovian Deterioration with Probability Bounds on Failure" Management Science Vol 9, 478-81 (1963)

28. Derman, C "Markovian Decision Processes--Average Cost Criterion" Mathematics of the Decision Sciences Vol 12, Part 2, 139-48 (1968)

29. Derman, C "On Sequential Control Processes" Annals of Mathematical Statistics Vol 35, 341-9 (1964)

30. Derman, C "Markovian Sequential Control Processes--Denumerable State Space" Journal of Mathematical Analysis Applications Vol 10, 295-302 (1965)

31. Derman, C and M Klein "Surveillance of Multi-component Systems: A Stochastic Traveling Salesman's Problem" Naval Research Logistics Quarterly Vol 13, 103-11 (1966)

32. Derman, C and GJ Lieberman "A Markovian Decision Model for a Joint Replacement and Stocking Problem" Management Science Vol 13, 609-17 (1967)

33. Derman, C and AF Vienott "Constrained Markov Decision Chains" Management Science Vol 19, 389-90 (1972)

34. Dirickx, YMI "Deterministic Dynamic Programming with Discount Factor Greater than One: Some Further Results and Algorithms" Zeitschrift Fur Operations Research Vol 18, 69-76 (1974)

35. Dirickx, YMI "Deterministic Discrete Dynamic Programming with Discount Factor Greater than One: Structure of Optimal Policies" Management Science Vol 20, 32-43 (1973)

36. Dirickx, YMI and MR Rao "Network with Gains in Discrete Dynamic Programming" Management Science Vol 20, 1428-31 (1974)

37. Eaton, JH and LA Zadeh "Optimal Pursuit Strategies in Discrete-State Probabilistic Systems" Transactions ASME, Series D; Journal of Basic Engineering Vol 84, 23-9 (1962)

38. Eppen, GD "A Dynamic Analysis of a Class of Deteriorating Systems" Management Science Vol 12, 223-40 (1965)

39. Finkbeiner, B and W Runggaldier "A Value Iteration Algorithm for Markov Renewal Programming" Computing Methods in Optimization Problems--2 (LA Zadeh, LW Neustadt and AV Balakrishman eds.) Academic Press (1969)

40. Fox, BL and DM Landi "An Algorithm for Identifying the Ergodic Subchains and Transient States of a Stochastic Matrix" Communications of the Association for Computing Machinery Vol 11, 619-21 (1968)

41. Geoffrion, AM "Elements of Large-Scale Mathematical Programming" Management Science Vol 16, 11, 652-91 (1970)

42. Grinold, RC "Elimination of Suboptimal Actions in Markov Decision Problems" Operations Research Vol 21, 848-51 (1973)

43. Grinold, RC "A Generalized Discrete Dynamic Programming Model" Management Science Vol 20, 1092-103 (1974)

44. Gustavson, FG "Some Basic Techniques for Solving Sparse Systems of Linear Equations" Sparse Matrices and Their Applications (DJ Rose and RA Willoughby eds.), Plenum Press (1972)

45. Gupta, SD "Use of Gain and other Estimates to Obtain Sub-Optimal Solution of a Class of Markov Decision Processes" International Journal of Control Vol 14, 6, 1031-40 (1971)

46. Hartman, JK and LS Lasdon "A Generalized Upper Bounding Method for Doubly Coupled Linear Programs" Naval Research Logistics Quarterly Vol 17, 411-29 (1970)

47. Hastings, NAJ and JMC Mello "Tests for Suboptimal Actions in Discounted Markov Programming" Management Science Vol 19, 1019-22 (1973)

48. Hastings, NAJ "Some Notes on Dynamic Programming and Replacement" Operational Research Quarterly Vol 19, 453-64 (1968)

49. Hastings, NAJ "Optimization of Discounted Markov Decision Problems" Operational Research Quarterly Vol 20, 499-500 (1969)

50. Hastings, NAJ "Bounds on the Gain of a Markov Decision Process" Operations Research Vol 19, 240-4 (1971)

51. Hastings, NAJ Dynamic Programming With Management Applications, Crane, Russak & Company (1973)

52. Hillier, FS and GJ Lieberman Introduction to Operations Research, Second Edition, Holden-Day (1974)

53. Hitchcock, DF and JB MacQueen "On Computing the Expected Discounted Return in a Markov Chain" Naval Research Logistics Quarterly Vol 17, 237-41 (1970)

54. Hordijk, A and HC Tijms "The Method of Successive Approximations and Markov Decision Processes" Operations Research Vol 22, 519-21 (1974)

55. Howard, RA Dynamic Programming and Markov Processes, Massachusetts Institute of Technology Press and John Wiley & Sons (1960)

56. Iglehart, DL "Dynamic Programming and Stationary Analysis of Inventory Problems" Multi-Stage Inventory Models and Techniques (H Scarf, Gilford and Shelly eds.) Stanford University Press (1963)

57. Jacques de Ruchet "How to Take Into Account the Low Density of Matrices to Design a Mathematical Programming Package" Large Sparse Sets of Linear Equations (JK Reid ed.) Academic Press (1971)

58. Jennings, A and AD Tuff "A Direct Method for the Solution of Large Sparse Symmetric Simultaneous Equations" Large Sparse Sets of Linear Equations (JK Reid ed.) Academic Press (1971)

59. Jewell, WS "Markov-Renewal Programming: I. Formulation, Finite Return Models" Operations Research Vol 11, 938-48 (1963)

60. Jewell, WS "Markov-Renewal Programming: II. Infinite Return Models, Example" Operations Research Vol 11, 949-71 (1963)

61. Jizmagian, GS "Generalized Programming Solution of Optimal Control Problems" Computing Methods in Optimization Problems--2 (LA Zadeh, LW Neustadt and AV Balakrishnan eds.) Academic Press (1969)

62. Kalymon, BA "Stochastic Prices in a Single Item Inventory Purchasing Model" Operations Research Vol 19, 1434-58 (1971)

63. Kalymon, BA "Machine Replacement with Stochastic Costs" Management Science Vol 18, 288-98 (1972)

64. Kalymon, BA "Structured Markovian Decision Problems" Naval Research Logistics Quarterly Vol 20, 1-11 (1973)

65. Karp, RM and M Held "Finite-State Processes and Dynamic Programming" SIAM Journal of Applied Mathematics Vol 15, 693-718 (1967)

66. Kaufmann, A and R Cruon Dynamic Programming-Sequential Scientific Management, Academic Press (1967)

67. Kemeny, JG and JL Snell Finite Markov Chains, Van Nostrand (1960)

68. Klein, M "Inspection-Maintenance-Replacement Schedules Under Markovian Deterioration" Management Science Vol 9, 25-32 (1962)

69. Klein, M "Markovian Decision Models for Reject Allowance Problems" Management Science Vol 12, 349-58 (1966)

70. Kolesar, PJ "Minimum Cost Replacement Under Markovian Deterioration" Management Science Vol 12, 694-706 (1966)

71. Kolesar, PJ "Randomized Replacement Rules Which Maximize the Expected Cycle Length of Equipment Subject to Markovian Deterioration" Management Science Vol 13, 867-76 (1967)

72. Kramer, JDR "Partially Observable Markov Processes" ScD Thesis, Department of Electrical Engineering, Massachusetts Institute of Technology (1964)

73. Kushner, HJ and AJ Kleinman "Numerical Methods for the Solution of the Degenerate Nonlinear Elliptic Equations Arising in Optimal Stochastic Control Theory" IEEE Transactions on Automatic Control Vol 13, 344-53 (1968)

74. Kushner, HJ and AJ Kleinman "Accelerated Procedures for the Solution of Discrete Markov Control Problems" IEEE Transactions on Automatic Control Vol 16, 147-52 (1971)

75. Kushner, HJ and AJ Kleinman "Mathematical Programming and the Control of Markov Chains" International Journal of Control Vol 13, 801-20 (1971)

76. Kushner, HJ and CH Chen "Decomposition of Systems Governed by Markov Chains" IEEE Transactions on Automatic Control Vol 19, 501-7 (1974)

77. Lasdon, LS Optimization Theory for Large Systems, The Macmillan Company, Collier Macmillan Limited, London, 234-42 (1970)

78. Lave, RE Jr "A Markov Decision Process for Economic Quality Control" IEEE Transaction Systems Science and Cybernetics SSC-2, 45-54 (19660

79. Lippman, SA "On Dynamic Programming with Unbounded Rewards" Management Science Vol 21, 1225-33 (1975)

80. MacQueen, JB "A Modified Dynamic Programming Method for Markovian Decision Problems" Journal of Mathematical Analysis Applications Vol 14, 38-43 (1966)

81. MacQueen, JB "A Test for Suboptimal Actions in Markovian Decision Problems" Operations Research Vol 15, 559-61 (1967)

82. Maitra, A "Dynamic Programming for Countable State Systems" Sankhyā Serial Vol A-27, 241-8 (1965)

83. Mandl, P "Analytic Methods in the Theory of Controlled Markov Processes" Transactions Fourth Prague Conference on Information Theory 1965, 45-53, Academia, Prague (1967)

84. Manne, AS "Linear Programming and Sequential Decisions" Management Science Vol 6, 259-67 (1960)

85. McCall, JJ "Maintenance Policies for Stochastically Failing Equipment: A Survey" Management Science Vol 11, 493-524 (1965)

86. Miller, BL "Dispatching from Depot Repair in a Recoverable Item Inventory System: On the Optimality of a Heuristic Rule" Management Science Vol 21, 316-25 (1973)

87. Morton, TE "Undiscounted Markov Renewal Programming via Modified Successive Approximations" Operations Research Vol 19, 1081-9 (1971)

88. Morton, TE "On the Asymptotic Convergence Rate of Cost Differences for Markovian Decision Processes" Operations Research Vol 19, 244-8 (1971)

89. Morton, TE "A Critique of the Norman-White Dynamic Programming Approximation" Operations Research Vol 17, 751-3 (1969)

90. Nakamura, M "On a Class of Stochastic Optimization Problems with a Specified Growth Pattern" Management Science Vol 20, 236-9 (1973)

91. Nakamura, M "Some Limit Theorems for a Class of Network Problems as Related to Finite Markov Chains" Journal of Applied Problems Vol 11, 94-101 (1974)

92. Norman, JM and DJ White "A Method for Approximate Solutions to Stochastic Dynamic Programming Problems Using Expectations" Operations Research Vol 16, 296-306 (1968)

93. Odoni, AR "On Finding the Maximal Gain for Markov Decision Processes" Operations Research Vol 17, 857-60 (1969)

94. Oliver, RM "A Linear Programming Formulation of Some Markov Decision Processes" presented at TIMS-ORSA, Monterey, California, April (1960)

95. Porteus, EL "Some Bounds for Discounted Sequential Decision Processes" Management Science Vol 18, 7-11 (1971)

96. Porteus, EL "On the Optimality of Structured Policies in Countable Stage Decision Processes" Research Paper No. 141 Revised, Stanford University (1974)

97. Reich, WJ, Flannery, WA and DA Miller "Reliability-Maintenability Cost Trade-Off via Dynamic and Linear Programming" Annals of Reliability and Maintenability Vol 5, 310-29 (1966)

98. Rolfe, AJ "Markov Chain Analysis of a Situation Where Cannibalization is the only Repair Activity" Naval Research Logistics Quarterly Vol 17, 151-8 (1970)

99. Rosenthal, RE "Stochastic Decision Processes in Location Analysis" PhD dissertation, School of Industrial and Systems Engineering, Georgia Institute of Technology (1975)

100. Rose, DJ and JR Bunch "The Role of Partitioning in the Numerical Solution of Sparse Systems" Sparse Matrices and Their Applications, (DJ Rose and RA Willoughby eds.), Plenum Press (1972)

101. Ross, SM Applied Probability Models with Optimization Applications, 150-152, Holden-Day (1970)

102. Rothstein, M "An Airline Overbooking Model" Transportation Science Vol 5 (1971)

103. Rothstein, M "Hotel Overbooking as a Markovian Sequential Decision Process" Decision Sciences Vol 5, 389-404 (1974)

104. Sackrowitz, H and E Samuel-Cahn "Inspection Procedures for Markov Chains" Management Science Vol 21, 261-70 (1974)

105. Satia, JK "Markovian Decision Processes with Uncertain Transition Probabilities or/and Probabilistic Observation of States" Technical Report No. 68-5, Department of Industrial Engineering, Stanford University, March (1968)

106. Satia, JK and RE Lave "Markovian Decision Process with Probabilistic Observation of States" Management Science Vol 20, 1-13 (1973)

107. Scarf, H "The Optimality of (S,s) Policies in the Dynamic Inventory Problem" Mathematical Methods in the Social Sciences (K Arrow, S Karlin and H Scarf eds.) Stanford University Press (1963)

108. Schweitzer, PJ "Multiple Policy Improvements in Undiscounted Markov Renewal Programming" Operations Research Vol 19, 784-93 (1971)

109. Shapiro, JF "Shortest Route Methods for Finite State Space Deterministic Dynamic Programming Problems" SIAM Journal of Applied Mathematics Vol 16, 1232-50 (1968)

110. Shapley, LS "Stochastic Games" Proceedings of the National Academy of Science USA Vol 39, 1095-100 (1953)

111. Sittler, RW "Systems Analysis of Discrete Markov Processes" IRE Transactions on Circuit Theory Vol CT-3, 257-66 (1956)

112. Smallwood, RD "Optimum Policy Regions for Markov Processes with Discounting" Operations Research Vol 14, 658-69 (1966)

113. Solberg, JJ "A Graph Theoretic Formula for the Steady State Distribution of Finite Markov Processes" Management Science Vol 21, 9, 1040-8 (1975)

114. Strauch, RE "Negative Dynamic Programming" Annals of Mathematical Statistics Vol 37, 871-90 (1966)

115. Taylor, HM "Markovian Sequential Replacement Processes" Annals of Mathematical Statistics Vol 36, 1677-94 (1965)

116. Totten, JC "Computational Methods for Finite State Finite Valued Markovian Decision Problems" Report ORC 71-9, Operations Research Center, University of California, Berkeley (1971)

117. Wagner, HM Introduction to Operations Research, Second Edition, Prentice-Hall (1975)

118. Wagner, HM "On the Optimality of Pure Strategies" Management Science Vol 6, 268-9 (1960)

119. Walsh, J "Direct and Indirect Methods" Large Sparse Sets of Linear Equations (JK Reid ed.) Academic Press (1971)

120. White, DJ "Dynamic Programming, Markov Chains, and the Method of Successive Approximations" Journal of Mathematical Analysis and Applications Vol 6, 373-6 (1963)

121. White, DJ "Dynamic Programming and Probabilistic Constraints" Operations Research Vol 22, 654-64 (1974)

122. White, JA "On Absorbing Markov Chains and Optimum Batch Production Quantities" AIIE Transactions Vol 11, 82-8 (1970)

123. White, LS "Markovian Decision Models for the Evaluation of a Large Class of Continuous Sampling Inspection Plans" Annals of Mathematical Statistics Vol 36, 1408-20 (1965)

124. White, LS "Bayes Markovian Decision Models for a Multiperiod Reject Allowance Problem" Operations Research Vol 15, 857-65 (1967)

125. Willoughby, RA "Sparse Matrix Algorithm and their Relation to Problem Classes and Computer Architecture" Large Sparse Sets of Linear Equations (JK Reid ed.), Academic Press (1971)

126. Wolfe, P and GB Dantzig "Linear Programming in a Markov Chain" Operations Research Vol 10, 702-10 (1962)

127. Young, D "Modeling Considerations" course notes, School of Industrial and Systems Engineering, Georgia Institute of Technology (1974)

128. Young, D "An Accelerated Method for Value Iteration" course notes, School of Industrial and Systems Engineering, Georgia Institute of Technology (1974)

129. Young, D and LE Contreras "Expected Present Worths of Cash Flows Under Uncertain Timing" Engineering Economist Vol 20, 257-68 (1975)

130. Young, DM Iterative Solution of Large Linear Systems, Academic Press (1971)

131. Zachrisson, LE "Markov Games" Advances in Game Theory (M Dreshner, LS Shapely and AW Tucker eds.) 211-53, Princeton University Press (1964)

132. Zaldivar, M and TJ Hodgson "Rapid Convergence Techniques for Markov Decision Processes" Decision Sciences Vol 6, 14-24 (1975)

VITA

Luis Eduardo Contreras, son of Mr. and Mrs. Eduardo Contreras Reyna, was born in Guadalajara, Jalisco, Mexico, on July 23, 1949. He was graduated from Instituto de Ciencias High School in Guadalajara, Jalisco, Mexico, in 1967.

He received the degree of Bachelor in Mechanical and Industrial Engineering from Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey, N.L., Mexico, in January 1972. As an undergraduate he received a scholarship and served as an instructor from January 1972 to August 1972.

Mr. Contreras entered the graduate program in the School of Industrial and Systems Engineering at Georgia Tech in September, 1972, having been awarded a United Nations Fellowship. He completed the requirements for the degree of Master of Science in Operations Research in September, 1973.

During his graduate program, Mr. Contreras served as a teaching and research assistant in the School of Industrial and Systems Engineering.

Mr. Contreras married the former Emilia Margarita Basave on June 16, 1972. Their first daughter, Cristina, was born January 11, 1974.